

Copyright

by

Wentao Fu

2012

The Thesis Committee for Wentao Fu
Certifies that this is the approved version of the following thesis:

**A Graph Grammar Based Approach
to Automated Manufacturing Planning**

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Matthew I. Campbell

Ata A. Eftekharian

**A Graph Grammar Based Approach
to Automated Manufacturing Planning**

by

Wentao Fu, B.E.

Thesis

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in Engineering

**The University of Texas at Austin
May 2012**

Acknowledgements

I would like to thank Dr. Matthew I. Campbell for his constant guidance, support and encouragement during the progress of this thesis and my graduate study in the US. It is his excellent research that ignited my enthusiasm in the research direction I am pursuing. Also I would like to thank Dr. Ata A. Eftekharian for serving as my second reader of my thesis and providing valuable feedbacks. This material is based upon the work supported by the Defense Advanced Research Projects Agency (DARPA) and I would thank DARPA as well. At the end, I am grateful for the support from my family and my girlfriend that always turns my dream into reality.

Abstract

A Graph Grammar Based Approach to Automated Manufacturing Planning

Wentao Fu, for M.S.E.

The University of Texas at Austin, 2012

Supervisor: Matthew I. Campbell

In this thesis, a new graph grammar representation is proposed to reason about the manufacturability of solid models. The knowledge captured in the graph grammar rules serves as a virtual machinist in its ability to recognize arbitrary geometries and match them to various machine operations. Firstly, a novel convex decomposition algorithm has been developed to decompose a given part into multiple sub-volumes, where each sub-volume is assumed to be machined in one operation or to be non-machinable. Then the decomposed part is converted into a graph so that graph grammar rules can determine the machining details. A candidate plan is a feasible sequence of all of the necessary machining operations needed to manufacture this part. If a given geometry is not machinable, the rules will fail to find a complete manufacturing plan for all of the sub-volumes. As a result of this representation, designers can quickly get insights into how a part can be made and how it can be improved based upon the feedback of the rules. A variety of tests of this algorithm on both simple and complex engineering parts show its effectiveness and efficiency.

Table of Contents

Table of Contents	vi
List of Tables	viii
List of Figures	ix
1. Introduction.....	1
2. Related Work	5
3. Convex Decomposition.....	8
4. Seed Lexicon.....	11
5. Rule Development	16
5.1 Choose Tool Entry Face.....	16
5.1.1 Accessibility Check	18
5.2 Choose Machine Type, Fixtures and Tool Type.....	21
5.3 Overall Description of the Design Process	28
6. Heuristic Search for Process Planning.....	31
7. Examples and Discussion	35
7.1 Description of Tested Cases.....	35
7.1.1 Validation of the Completeness of the Representation Space	36
7.1.2 Non-Traditional Manufacturing Processes in the Representation	38
7.1.3 Heuristics in the Representation for Feature Interactions.....	41
7.2 Characteristics of Implementation and Deployment.....	45
7.3 Manufacturing Processes Handled.....	47
7.4 Part Geometries That Can Be Processed	49
7.5 Validation.....	53

8. Conclusion and Future Work	60
References	62
Vita	64

List of Tables

Table 1:	The list of all the labels used in the seed lexicon.....	15
Table 2:	The pseudo code describing the accessibility check in case 3	21

List of Figures

Figure 1:	Flowchart of the Representation Scheme	2
Figure 2:	Convex Decomposition Tree	9
Figure 3:	The simple solid model shown as part A in Figure 1 as a seed graph in representation	11
Figure 4:	A sample face representation in the seed lexicon	12
Figure 5:	An example of infeasible tool entry face in case 1	18
Figure 6:	An example of infeasible tool entry face in case 2	19
Figure 7:	An example of infeasible tool entry face in case 3	20
Figure 8:	The right view of the negative solid shown in Figure 7(b).....	21
Figure 9:	The VMC rule from Rule set #5 in the grammar representation	22
Figure 10:	Screenshots of two drilling rules in GraphSynth	24
Figure 11:	Several parts showing the applicability of end milling and ball milling	25
Figure 12:	Screenshots of three milling rules in GraphSynth	27
Figure 13:	Flowchart of rule sets	29
Figure 14:	A sample manufacturing plan for the solid model shown as part A in Figure 1	36
Figure 15:	Summary report of the search space for the part discussed in Figure 1	37
Figure 16:	The main chassis part.....	39
Figure 17:	A sample manufacturing plan for the chassis part	40
Figure 18:	The solid model of radio box	42
Figure 19:	A sample manufacturing plan for the radio box	43

Figure 20:	AMFA GUI Web Page.....	46
Figure 21:	Sample part 1—the Radiobox.....	49
Figure 22:	Sample parts 2 and 3	50
Figure 23:	Sample parts 4.....	51
Figure 24:	Sample part 5—A pre-formed Bradley Vehicle Bracket.....	52
Figure 25:	Sample part 6—A pre-formed chassis part.....	52
Figure 26:	A new manufacturing plan for the pre-formed chassis part.....	53
Figure 27:	A sample manufacturing plan generated from featureCAM for part A in Figure 1	54
Figure 28:	A sample manufacturing plan generated from the representation for part A in Figure 11	57
Figure 29:	A sample manufacturing plan generated featureCAM for part A in Figure 11	57
Figure 30:	The comparison between the original part A in Figure 11 and the solid machined in featureCAM.....	58

1. Introduction

In order to better streamline the interaction between the mechanical design process and manufacturing, an approach is being developed which automatically reasons about CAD models to define detailed and optimal manufacturing plans. The larger system is known as AMFA: Automated Manufacturing Feedback Analysis (AMFA), and this paper presents the graph grammar based representation scheme that serves as the system's foundation. Starting from a CAD model provided by the designers, the grammar representation scheme performs an analysis of manufacturability on it by referencing the data provided on the manufacturing facility that will build the part in question. The outputs of AMFA are potential manufacturing plans, their associated times and costs, and, in certain cases, recommendation on how to change the part so that it is more easily manufactured.

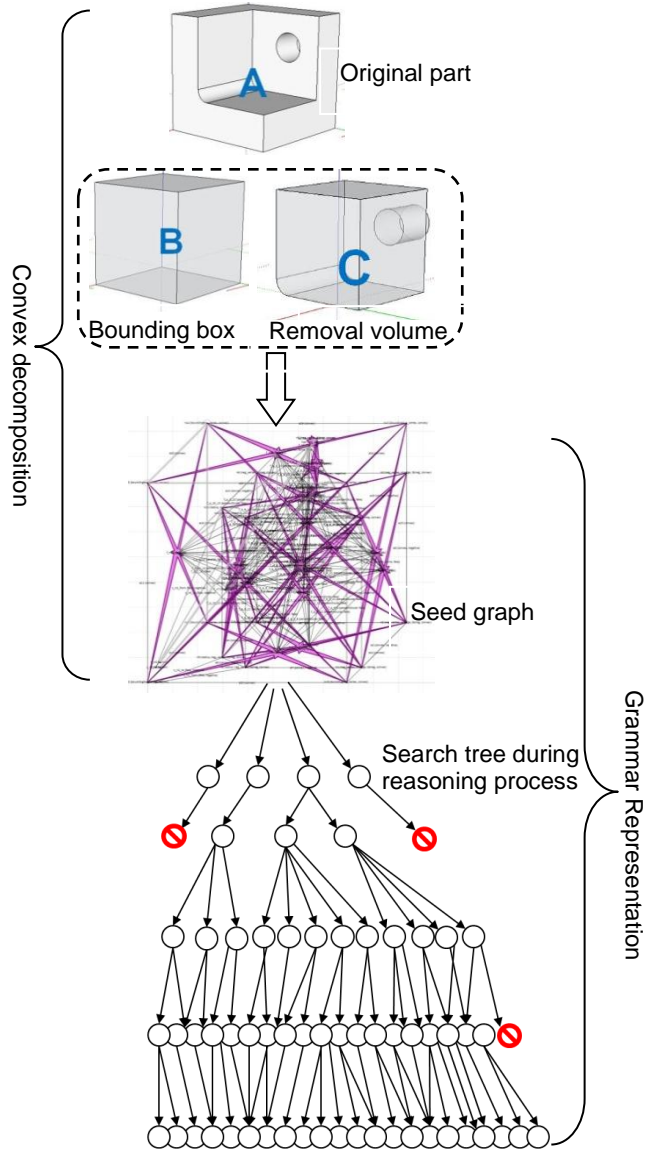


Figure 1: Flowchart of the Representation Scheme

To generate the outputs, two distinct areas of efforts are involved in the representation scheme. They are summarized into the Convex Decomposition and the Grammar Representation as shown in Figure 1. Firstly, a new method of decomposing 3D solid models was developed, which is indicated in the upper portion of Figure 1. To

begin with, a CAD model in the STEP format¹ (Part A in Figure 1) provided by designers is loaded into the tool. Then this geometry – comprised of vertices, edges, and faces – is parsed into a label-rich graph which serves as the basis for the representation (the lower portion in Figure 1). In the translation, a bounding box (Part B in Figure 1) is extrapolated from the original part A since one would likely start the actual manufacturing from a larger block of material. The original part A is then subtracted from the bounding box to create the removal volume (Part C in Figure 1) that is to be removed. This removal volume then undergoes further divisions to generate compact sub-volumes where each sub-volume is assumed to be machined in one operation or to be non-manufacturable. The decomposed removal volume is then converted into a graph and used as the initial seed in the representation.

The grammar representation reasons about the manufacturability of a given part under certain foundry capabilities. Firstly, all available manufacturing processes within a foundry are translated into grammar rules. The rules are then organized to reason about the seed graph in order to determine its machining details. A search tree is drawn to describe how they work on the seed graph. Steps in the tree represent alternative manufacturing operations for different sub-volumes. These operations are determined through the rules which detect a series of graph elements and relate them to a particular manufacturing process. Each operation consists of the tool entry face, the tool type choice, the machine choice, and the needed fixture to machine one sub-volume. As the tree grows, more and more sub-volumes get manufactured. When the tree propagates to its bottom, there are no more sub-volumes of the given part to be machined, and a complete search space that includes all alternative manufacturing plans for the given part

¹ .STEP is used as the standard format for the exchange and conversion of solid models

is derived. In addition, by translating foundry capabilities into graph grammar rules, a precise conclusion of non-manufacturability of a part can be made if the rules fail to find a manufacturing plan for a part. This signals that the manufacturing process is beyond the foundry capability and this part needs to be redesigned.

This work describes the grammar representation and shows how the method functions on a few test parts. In the next section the relevant research is described and how it compares to this project (Section 2). This is followed by a description of how the STEP file is converted into the seed graph for use in the Grammar Representation (Section 3). Sections 4 and 5 present the seed format and the rules that reason with the graph. In Section 6, a tree search algorithm is described which used the representation. The thesis closes with examples and discussion (Section 7) and a conclusion and future work section (Section 8).

2. Related Work

Automated manufacturing planning was first proposed by Russel [1] in 1967. Due to the fast development of Computer-Aided Process Planning (CAPP) since the 1980s, this topic continues to receive much attention both from researchers and practitioners [2]. In this time, many knowledge based approaches [3, 4, 5] were developed to capture the basic logic used by a process planner. Marri, Gunasekaran and Grieve [2] provided a comprehensive review of these CAPP based systems. Based on their conclusions, more attention should be paid to the architecture and constraints for machining operations while developing a CAPP system. More recently, Sharma and Gao [6] proposed a process planning system using the latest tools and technologies and to fully comply with the international standard for exchange of product data, but it is only intended for simple prismatic parts and the feedback analysis cannot be automatically imported into CAD systems for detailed redesign. Allen et al. [7] developed an agent-based approach that can provide a number of generic solutions whilst maintaining the ability for manual intervention to establish local working preferences, but the efficiency of this algorithm is restricted by its parametric optimization process.

In contrast, the graph grammar based approach to automated manufacturing planning considers a large variety of topologies rather than being restricted by the optimization process. It utilizes a technique of creating new graphs from an original graph (host) by applying prescribed rules onto the host [8]. The rules are of the form $L \rightarrow R$ where the left hand side (LHS) includes elements and conditions to be recognized and satisfied in the host graph and the right hand side (RHS) indicates the transformations of those elements that have been recognized in the host.

A widely used grammar based approach in automated manufacturing planning is Form-Feature Recognition (FFR) technique [9]. This is primarily important because it can extract or generate higher level and meaningful geometric entities that are not easily inferable from the solid geometry. These geometric features can serve as a bridge between the geometry on one hand and manufacturing reasoning on the other hand which is a crucial step in computer-aided part design [9]. A thorough survey of various techniques in Form-Feature Recognition can be found in the work by Han [10], Shah [11, 12], and Subrahmanyam [13]. According to these surveys, three dominant techniques which include graph, volumetric and hint based approaches are mostly used in modern FFR algorithms. Graph based techniques, although proven to be reliable in recognizing isolated features, suffer from the complexity of the geometry and the fact that features may have interactions with each other [14, 15]. Some researchers [16] have tried to tackle the problem by introducing various types of heuristics to the algorithm and have gained considerable achievements but still the problem remains unsolved for complex geometries. Others [14, 17] have tried to add missing elements that correspond to interacting features into the graph but despite the added complexity they do not completely solve the problem.

Volumetric decomposition methods stand apart from the others, both in the algorithm employed and the results. Researchers have continued to extend and refine this approach to solve numerous shortcomings, such as non-convergence and geometric domain restrictions. The volumetric decomposition method can handle interactions and provide additional information such as geometry-based precedence relations [18]. A very similar approach to that proposed in this paper was provided by Ertelt and Shea [19]. Knowledge of fundamental machine capabilities was encoded by generating a vocabulary of removal volume shapes based on the available tool set and machine tool motions.

Since this method coupled the topological representation and parametric evaluation process together, its scalability to complex parts and non-traditional machined parts is questionable.

3. Convex Decomposition

In the context of computer-aided manufacturing, convex decomposition is primarily important and useful for generating simple removal volumes from the work-piece. These topological entities are commonly referred to as machining features in literature. In this work we extend the idea beyond the volumetric decomposition approach for 3D solid models by adding a layer of reasoning to the algorithm. Our decomposition algorithm [20] uses a concave-edge ranking strategy to prioritize division and two sets of heuristics to evaluate the direction of cut within each division. Results of convex decomposition can be represented as a tree structure with branching factor equal or greater than 4 (4 is the case when there are exactly two solids generated after each cut) and depth of the tree equal to the total number of concave edges in the solid.) . Each node represents a volume that needs to be cut and each branch represents a left (L) or right (R) cut in the tree. Each nodes consists of either a simple shapes (i.e. a convex volume) which is represented as an (S) in the tree or a complex shape (i.e. a volume with one or more concavities) which is represented as a (C) in Figure 2.

For a given solid model as shown on the top of Figure 2, let us consider a case where the left branch is always chosen as the preferred cut, and the result after first cut is one complex (C) volume and one simple (S) volume. The simple volume does not need any further cutting operations so the branch related to this node stops here; this is shown as a red node in the tree. For each complex volume (C) the branch grows further to lower levels until it hits a simple volume (S). In order to find the best result (or an optimal decomposed solid for the representation) one can generate all the possible options in the tree and explore and evaluate each individual option from the pool of solutions. This, however, is not a good idea due to the fact that Boolean operations are computationally expensive and slow in CAD kernels. Furthermore evaluation conceptually requires a

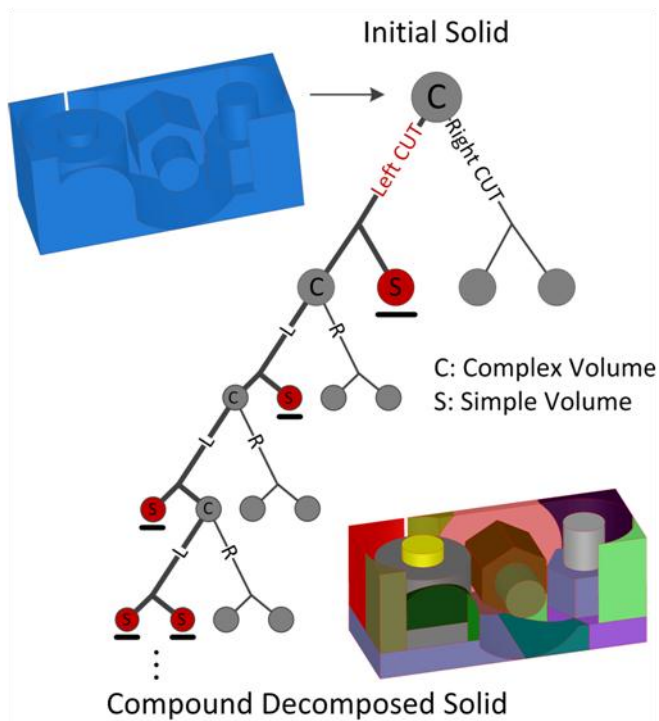


Figure 2: Convex Decomposition Tree

human to inspect the result and decide if it is a good decomposition or not (computationally evaluating the quality may be possible, but it is out of the scope of the work). Therefore, it is nearly impossible to explore all the options or branches in the tree. The alternative solution is to expand a single but promising branch. At each level in the tree the algorithm evaluates the solutions and decides the preferred direction for the next level. This continues until no more complex volumes are detected. In order to reach this goal, two sets of heuristics are designed and implemented to guide the convex decomposition algorithm along the desired direction.

It is important to note that a desired solution is described as a decomposed volume that contains partitions which are suitable for manufacturing purposes. In other words, all the sub-convex-volumes should possess certain properties which include (1) having a compact shape with no or few number of concavities, (2) having a prismatic or close to prismatic geometry. Due to the lack of space, details about the algorithm, the procedure for cutting the solid and evaluating each cut are omitted here and we refer the interest reader to [20]. Based on the designed heuristics, only desirable branches will survive and continue to grow until no further volumes of type C are recognized by the algorithm. The final decomposed shape is a combination of remaining S type volumes in all traversed branches. The candidate decomposed solid at the bottom of Figure 2 is a sample result after the heuristic-guided convex decomposition.

4. Seed Lexicon

After the removal volume of a given solid model is decomposed, the compound solid comprised of different sub-volumes has to be translated into a seed graph such that the grammar representation can work on it. Rather than using existing graph techniques to represent a solid model, a new lexicon is proposed. Figure 3 gives an example showing that how a solid model shown as part A in Figure 1, is described as a label-rich graph.

This example is a simple shape with a pocket in front and a through hole in the back. In this graph, geometric elements are described by nodes, arcs, and hyperarcs. Nodes are used to represent vertices and faces. Arcs are used to represent edges in the shape as well as to indicate relative positioning information (parallel, perpendicular, etc.) between any two faces. For example, if two faces are parallel, the arc connecting the face nodes of these two faces will have a label “parallel”. A hyperarc is a special arc. While

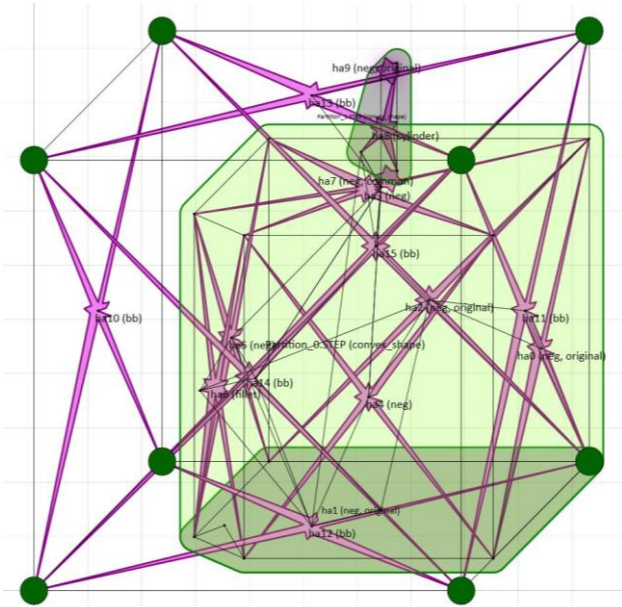


Figure 3: The simple solid model shown as part A in Figure 1 as a seed graph in representation

arcs can only connect two nodes, a hyperarc can connect as many nodes as needed. It is always used to connect all vertices belonging to a face to their face node. Figure 4 gives an example of a complete representation for a face with four vertices in the seed lexicon. The node n0 with label “face” is to represent the face and the other four nodes with label “neg_vertex” are used to represent all the vertices. They are connected by hyperarc 0, which also has a label “face”. This label is to distinguish this hyperarc as a face hyperarc from the other types of hyperarcs used in the lexicon. A complete roster of all the labels used in this lexicon is tabulated in Table 1 for reference.

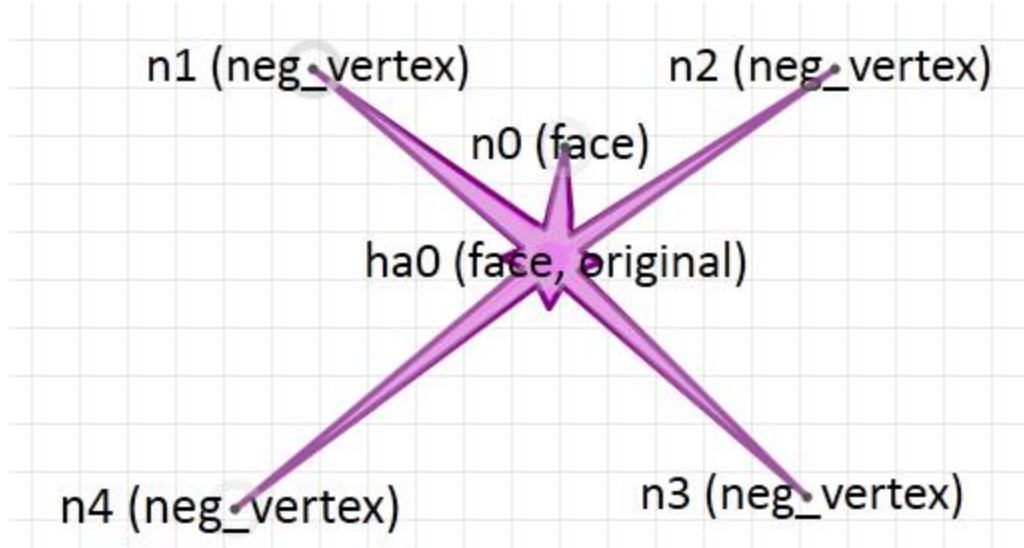


Figure 4: A sample face representation in the seed lexicon

Another type of hyperarc is designed to encompass a sub-volume by connecting all of the nodes of a sub-volume together. For example, in Figure 3, the hole and the bottom cuboid are separated by two green hyperarcs. By using nodes, arcs, and hyperarcs in the seed graph, all geometric information about vertices, edges and faces for a solid model is stored. In this way, face nodes are used in the seed and rules to refer to general machining features, like holes, pockets and slots, if applicable. Moreover, edges and

vertices provide more details about shapes and geometries, which are necessary for the rules to describe manufacturing operations more precisely.

It is also important to note the variety of labels in the graph, which are used to store topological, rather than parametric, information. Table 1 selectively tabulates the labels used in the seed lexicon. For example, a face node may have label “bb”, which indicates that this face is a bounding box face. If a face node represents a face belonging to the removal volume, it will have the label “neg”. In Figure 3, the face node n1 (not explicitly shown, but overlaps with its face hyperarc ha1), which represents the bottom face of the cuboid, has label “neg”, while face node, n12, has label “bb. A hyperarc may have label “original”, which means the face this hyperarc connects to is an accessible face to the tool and it is a candidate face for the tool entry face selection. Examples include the hyperarc ha1 and ha9 in Figure 3 and ha0 in Figure 4. Besides, the face adjacency property, either “convexity” or “concavity”, between any two adjacent faces is stored in the label of their common edge. With these labels, the rules can do a much more precise reasoning about the graph elements they capture and the search can be more successful. The detailed explanation of the rules based upon the label-rich seed graph is given in section 5.

Geometrical element in the seed	Description	Related Labels	Explanation
Face node	Represent a face	machining_start	This face is chosen as a tool entry face
		common	This face is shared by two or more sub-volumes
		face	Indicate this node represents a face
		bb	This face is a bounding box face
		neg	This face is a surface of the negative solid
		planar	This face is a planar face
		non_planar	This face is not a planar face
		fillet	This face is a fillet face
		cylindrical	This face is a cylindrical face
		machined	This face is machined
		fixed	This face is fixed
Face arc	Connect two faces	parallel	Two faces are parallel
		perpendicular	Two faces are perpendicular
Face hyperarc	Connect together all elements of a face	original	Indicate the face this hyperarc represents is accessible
			Indicate the geometric element this hyperarc represents is of type face

Table 1: The list of all the labels used in the seed lexicon

Edge arc	Represent an edge	tangential	Indicate the two adjacent faces this edge belongs to are tangential to each other
		convex	The two adjacent faces this edge belongs to are convex to each other
		concave	The two adjacent faces this edge belongs to are concave to each other
		common	This edge is shared by two or more sub-volumes
		accessible	This edge is accessible to the tool
		curved	This edge is not a linear edge
Vertex node	Representing a vertex	onedge	This vertex is on an edge of bounding box
		common	This vertex is shared by two or more sub-volumes
		neg_vertex	This vertex is a vertex of the negative solid
		boundingbox_vertex	This vertex is a vertex of the bounding box
Sub-volume hyperarc	Representing a sub-volume by connecting all elements of a sub-volume together	convex_shape	Indicate that this hyperarc refers to a sub-volume
		current_shape	Indicate that the sub-volume this hyperarc represents is the current sub-volume that is being machined
		machined	Indicate this sub-volume has been machined (removed)

Table 1 (continued): The list of all the labels used in the seed lexicon

5. Rule Development

With an understanding of how a compound solid model comprised of different sub-volumes for a given part is represented in the seed, it becomes easier to tell how the rules are developed. In this part, eight sets of graph grammar rules have been devised to simulate a virtual machining process, removing material of the compound solid step by step. This process will end if the volume of the compound solid becomes zero. These rule sets are arranged in a specific sequence such that they collectively perform the required reasoning as a whole.

The first rule set (rule set 0), the “pre-processing” rule set, aims to recognize typical sub-volumes (counter-sink, round edge, etc.) and non-traditional machining operations (bending, etc.) which are tagged for later use. These sub-volumes are usually machined in a finishing process using specific tools. By recognizing and isolating these special cases at the first stage, more realistic manufacturing plans which separate roughing and finishing processes can be generated. Unlike other machining operations, bending operations do not remove material. They simply change the shape of the seed graph. However, this change does affect the generation of a correct bounding box for a given part. In this case, the rules in this rule set operate in cooperation with the Convex Decomposition to implement these non-material-removal operations on the part before a correct bounding box is generated.

5.1 CHOOSE TOOL ENTRY FACE

The second rule set (rule set 1) is to choose a sub-volume of the negative solid that is going to be removed. The candidate sub-volume is chosen among all the non-machined sub-volumes of the negative solid. After a sub-volume has been chosen, this sub-volume’s hyperarc will get a label “current_shape”. Technically the options this rule

set recognizes should be the same as the total number of sub-volumes that have not been machined. However, because the inner sub-volumes are always encompassed by the adjacent outer sub-volumes, they become accessible only after all of the outer sub-volumes are removed. As a result, only the outer non-machined sub-volumes are actually accessible at this time. Therefore, the concurrent feasible options for this rule set are greatly reduced. The following rule set (rule set 1) will terminate the search loop right away once an inaccessible sub-volume is chosen in this rule set.

The third and fourth rule sets (rule set 2 and 3) are used to identify a feasible tool entry face on the sub-volume that is chosen from rule set 1 and is to be machined. Firstly, in rule set 2, a single rule is designed to capture a face which is accessible by the tool. If such a face is found, it will be labeled with “machining_start”. If no faces are found, the process terminates – the sub-volume chosen in rule set 1 is currently inaccessible. In rule set 3, there are three rules, representing three special cases, where a tool entry face selected from rule set 2 needs to be rechecked. For case 1, an accessible face is not eligible to be a tool entry face for the current sub-volume if this face itself is part of a larger accessible face and the larger face is also a candidate tool entry face for the current sub-volume. For the decomposed part shown in Figure 5, there are three adjacent sub-volumes. The blue and purple sub-volumes on the top have to be removed first before the yellow sub-volume on the bottom can be removed. In this case, the middle common face 3 shared by the two upper sub-volumes and the yellow sub-volume is decomposed into two smaller common faces 1 and 2. The second start face check rule will ensure that only the entire face 3 is chosen as the tool entry face and both face 1 and face 2 will be excluded since they both constitute only a part of face 1.

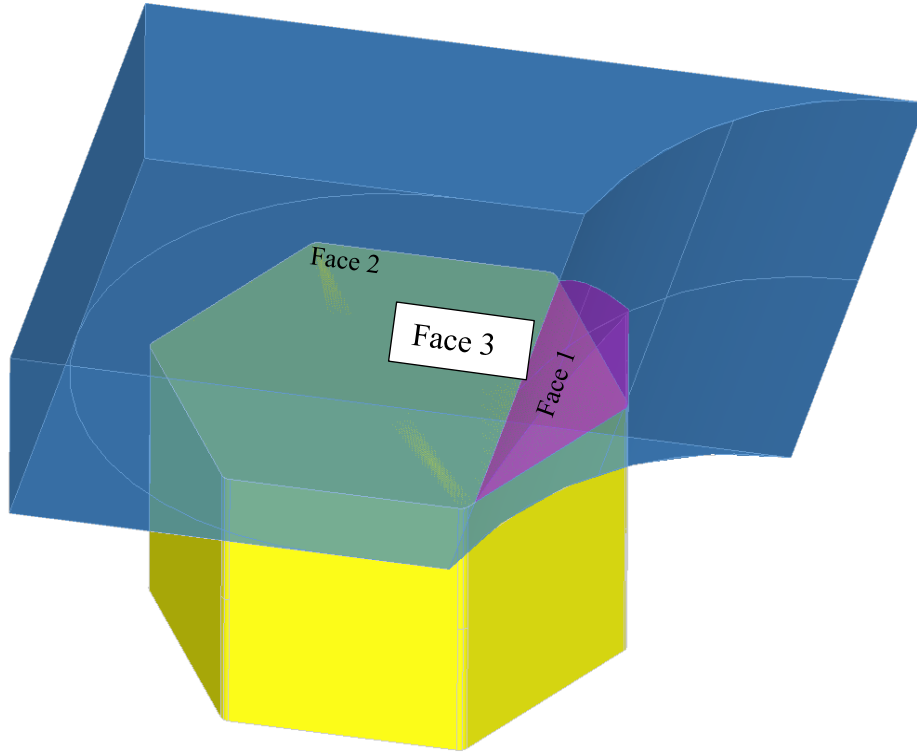


Figure 5: An example of infeasible tool entry face in case 1

5.1.1 Accessibility Check

Both case 2 and case 3 in rule set 3 implement the accessibility check of the current sub-volume from the tool entry face selected in rule set 2. A tool entry face is valid if the sub-volume is fully accessible from this face. Otherwise, the two cases in rule set 3 will invalidate the face selected in rule set 2 since the current sub-volume cannot be fully removed from this face, which violates the basic assumption under this representation tool that each sub-volume must either be removable in one single operation, or be non-machinable.

For case 2, an accessible face is not allowed to be chosen as a tool entry face if it is part of a larger but inaccessible face. An infeasible tool entry face is described in

Figure 6 using the same solid model as discussed in Figure 3. If the hole is first removed, the internal circular face of the hole that is shared by the front pocket becomes accessible to the tool. But this face is not a good choice of the tool entry face since from this face the tool cannot access the whole material of the pocket. Despite being a valid face to begin machining in rule set 2, the choice is invalidated in rule set 3.

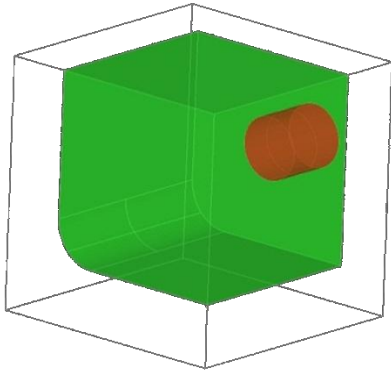


Figure 6: An example of infeasible tool entry face in case 2

For case 3, an example part is provided to illustrate the scenario where a tool entry face is not a valid option to remove the entire sub-volume. The original solid was provided by ASU and is given in Figure 7(a). The compound negative solid consisting of all decomposed sub-volumes is shown in Figure 7(b). Interestingly, there is a beam-shaped sub-volume (the green shape) laying on top and going across the entire length of the negative solid. If the tool enters from the top and feeds downward (indicated as the orange arrow), then without removing the green beam, all the transverse sub-volumes that this beam sits on are not fully removable since the beam blocks a certain amount of material that the tool cannot access for all the four sub-volumes (from left to right: the dark yellow, gray, dark blue, and dark pink sub-volumes).

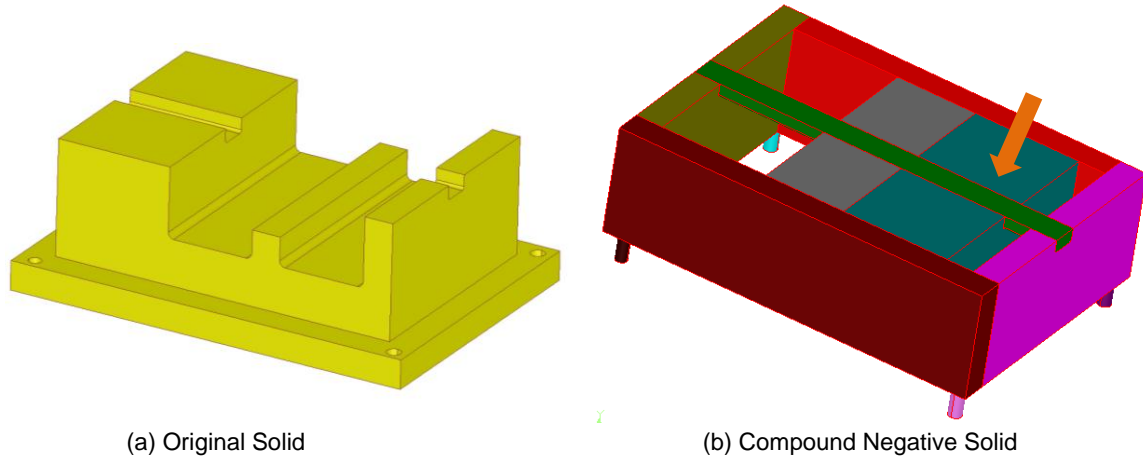


Figure 7: An example of infeasible tool entry face in case 3

In order to implement the accessibility check described in case 3, the face properties of the current sub-volume and their relation with the tool entry face need to be investigated. Figure 8 gives the right view of the negative solid shown in Figure 7(b). The tool feed direction, the tool entry face, and face i , which is the face that the tool cannot access from the tool feed direction, are indicated. Notice that for the current sub-volume, the face normal for each face always points outward from the sub-volume. By inspection, it is not hard to find that only the face normal of face i has a negative inner product with the tool feed direction. The other normals have either positive, or zero inner product with the tool feed direction. This observation remains valid for all the other cases that have been investigated. Therefore, it is safe to sum up a necessary condition for the accessibility check in case 3, which is expressed in pseudo code in Table 2.

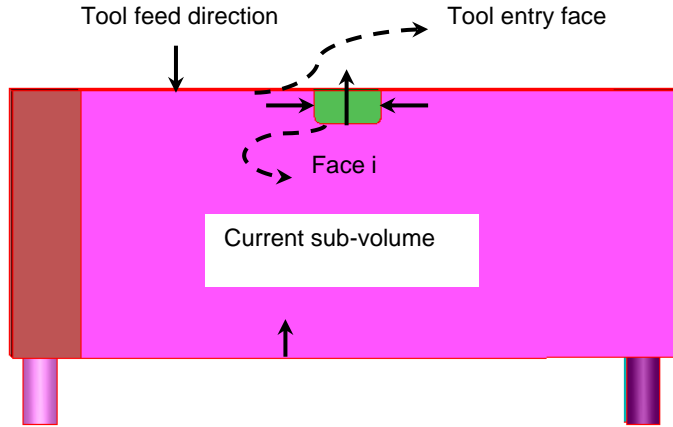


Figure 8: The right view of the negative solid shown in Figure 7(b)

Note that the face i should not be accessible in this accessibility check. If face i is accessible, meaning that the green beam-like sub-volume has been removed, then the dark pink sub-volume is actually removable from the tool feed direction.

<p><i>Given the tool entry face and the tool feed direction,</i></p> <p><i>For every other face i in current sub-volume</i></p> <p><i>If</i></p> <p><i> Inner product (face i, tool feed direction) < 0 && face i is not accessible</i></p> <p><i>Then</i></p> <p><i> Current sub-volume is not fully accessible from the tool feed direction;</i></p> <p><i> Invalidate the tool entry face;</i></p> <p><i> Break;</i></p> <p><i>End</i></p> <p><i>End</i></p>
--

Table 2: The pseudo code describing the accessibility check in case 3

5.2 CHOOSE MACHINE TYPE, FIXTURES AND TOOL TYPE

After rule set 2 and 3, a feasible tool entry face has been identified. Then, the representation goes to rule set 4, which is used to choose the machine type used to

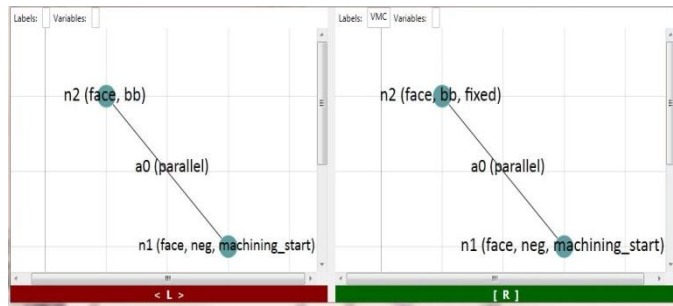


Figure 9: The VMC rule from Rule set #5 in the grammar representation

manufacture the current sub-volume and to define the corresponding fixture method. Currently these two decisions are made in one single rule by considering which bounding box face to fix and the relative positioning relation between the fixed face and tool entry face of this sub-volume. If they are parallel, then a VMC, or Vertical Machining Center, is chosen. If they are perpendicular, a HMC, or Horizontal Machining Center, is recognized. The rule for the VMC is shown in Figure 9. The LHS of this rule captures a pair of parallel faces while one face is a bounding box face (the face node n2 labeled with “face” and “bb”) and the other face is a tool entry face (n1 labeled with “face”, “neg” and “machining_start”). If these two faces are found, then the bounding box face n2 will be “fixed” and the “VMC” will be chosen. Additionally, if the current sub-volume is detected as a sheet metal part, the water jet will also be invoked through a specific rule in this rule set. Future endeavors for this rule set are focusing on extending machine types to multi-axis machines such that more general fixtures can be covered.

After rule set 4, the current sub-volume has been fixed in a machine and the tool entry face for this sub-volume has also been determined. Then, the representation goes to rule set 5, which is responsible for tool type selection. Rule set 5 is a cluster of available tooling operations, including Drilling, Milling (End Milling and Ball Milling), Sheet Cutting (Water Jet), and Counter-sinking. Each tooling operation corresponds to one or

more rules in this rule set. These rules are specially designed based on how each operation is implemented.

For example, in this rule set there are two drilling rules as shown in Figure 10. The reason for creating two rules is that there are two different representations for holes in STEP files. The planar circular face of a hole can be represented with either two vertices and two semi-circular edges (type 1) or one vertex and one 360° circular edge (type 2). Figure 10(a) recognizes the first hole type. The LHS of this rule finds a hole to be machined by capturing its cylindrical face (a hyperarc labeled with “cylinder”) and one of its planar faces, which is accessible by the tool and is depicted as another hyperarc labeled with “machining_start”. Additionally, since this hole is also a sub-volume to be machined, its sub-volume hyperarc (a hyperarc with label “convex_shape”) is also captured. If such a hole is found, a drilling operation is implemented on this sub-volume, which is described by a virtual transformation from LHS to RHS of the rule. After that, this sub-volume’s machining_start face and its sub-volume hyperarc become “machined”. Similarly, Figure 10(b) is the drilling rule for the hole of type 2.

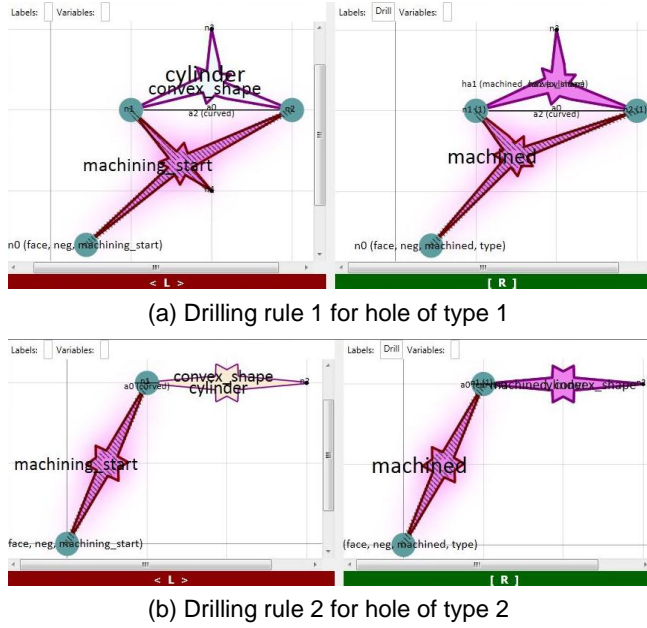


Figure 10: Screenshots of two drilling rules in GraphSynth

However, the milling operations are much trickier to capture in the rules due to the wide applicability of the milling tools. For the end milling, it is intended for use in most cases, such as profile milling (pockets, holes, slots, chamfers, etc.), face milling, tracer milling or plunging. It has cutting teeth at one end, as well as on the sides. While all the cutting teeth can cut off materials when the tool is feeding, the side teeth can also be used to mill out desired fillets (Figure 11, part b). But the bottom teeth are not capable of creating fillets with high tolerance (Figure 11, part c). Ball milling, on the contrary, is applicable when there are lots of fillets that are to be created (Figure 11, part c). But it is poor at preserving sharp edges since it will always leave fillets around these edges. The same problem arises for the end mill tool if there is any sharp edge along the feed direction of the tool (Figure 11, part a). To capture all the constraints above, three rules are designed as shown in Figure 12.

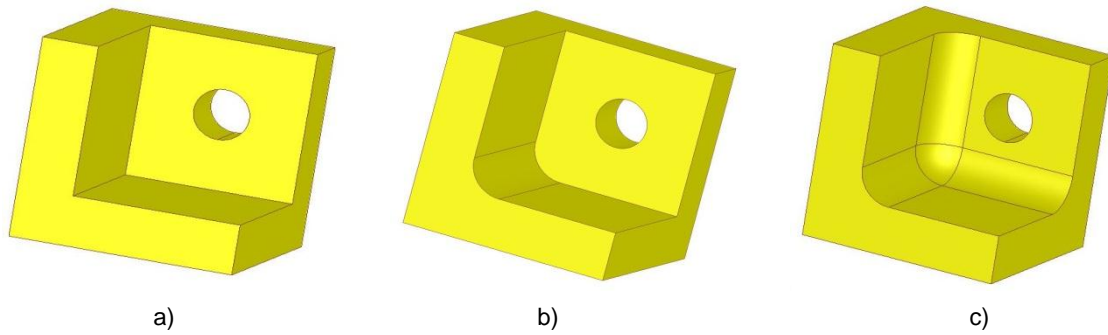


Figure 11: Several parts showing the applicability of end milling and ball milling

In Figure 12, the first rule (from the top) is a trigger rule. It is a first level check of the manufacturability of a given part in a milling operation. On the left hand side, the rule first finds the tool entry face (face node n_0 with labels “face” and “machining_start”) selected in rule set 2, then detects the existence of any inaccessible sharp edge (arc a_0 connecting vertices n_1 and n_2). If such kind of edge is found, the rule will terminate the search process, which means the part fails to pass the first level check and it is not machinable in milling operations. For example, part a in Figure 11 will be excluded by this rule since there is an inaccessible sharp edge that the milling tools cannot create from any tool entry face. If this sharp edge is not found, the rule will tag a global label “Mill_cand” in the seed graph representing the part. The second and third rules will only be called if the seed graph passed to them has this “Mill_cand” label. Each of these two cascading rules imposes a second level check of the manufacturability of the given part in the specific milling operation that this rule represents.

For example, the second milling rule in the middle of Figure 12 is a second check for the end milling operation. It verifies that for all the faces that are to be created, there is no desired fillet that needs to be milled out using the bottom cutting teeth of the tool. Under this constraint, part b in Figure 12 is machinable using the end mill tool, but part c is not since in order to remove the front cuboid, the bottom teeth of the tool have to create

desired fillets from any possible tool entry face. However, for the ball mill tool, it is the opposite scenario. The ball mill tool is able to machine the part in Figure 12 c), but not the part in Figure 12 b). The reason is that the ball mill tooling operation will leave an unwanted fillet along any sharp edge it machines. To implement this constraint, the third milling rule (bottom of Figure 12) is designed. The left hand side of the rule does the same logic reasoning as the second check rule for the end mill tool except that there should be no sharp edges in any face that is to be created by the ball milling. If all the conditions specified on the LHS are satisfied, this rule will change the global label of the seed graph from “Mill_cand” to “Ball_Mill”, representing that the ball milling operation is implemented on the current sub-volume of this part.

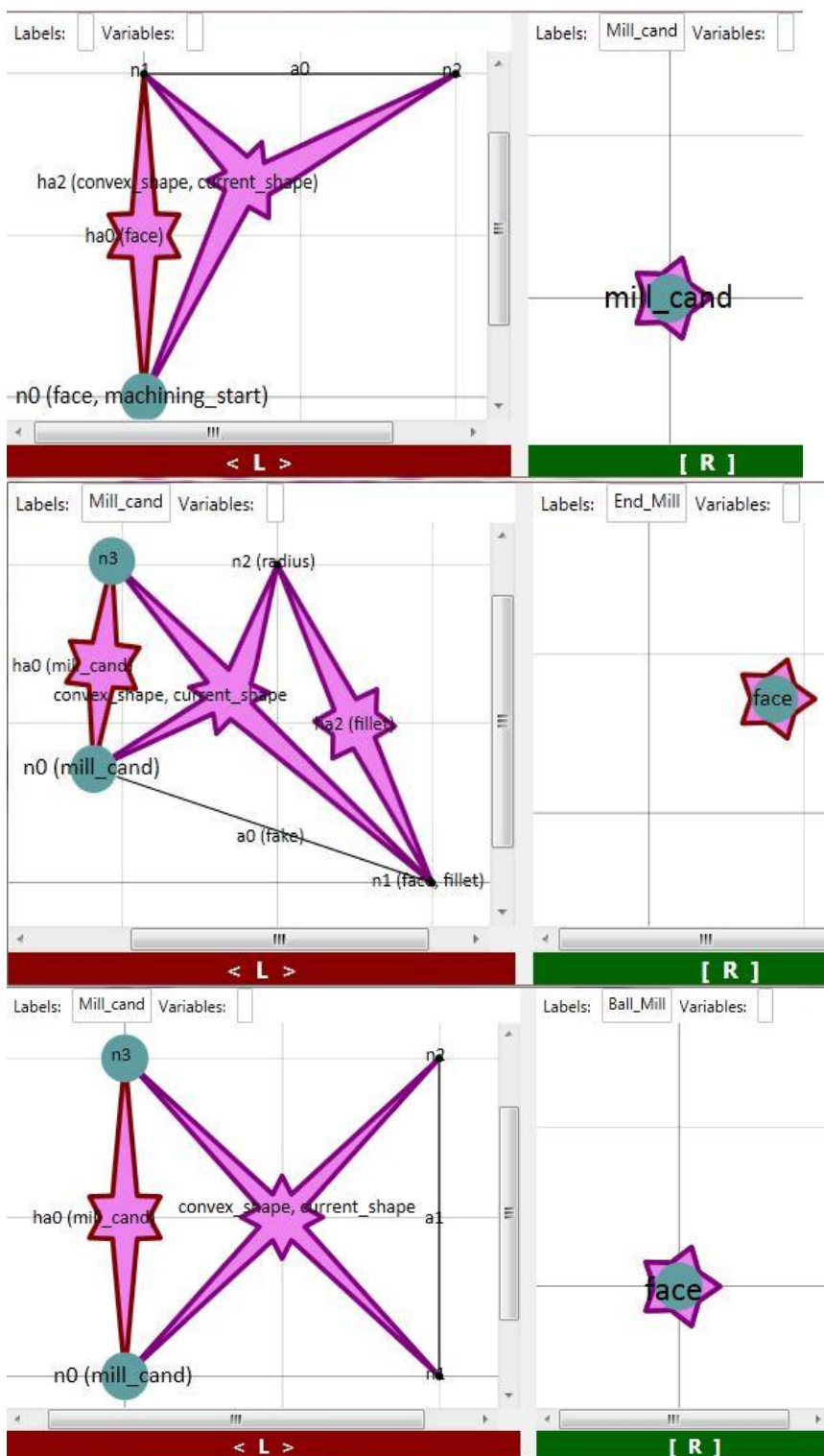


Figure 12: Screenshots of three milling rules in GraphSynth

Similar algebraic reasoning is performed for the remaining tool types where complete Left Hand Sides have been developed to capture the intricacies of the geometric constraints and the Right Hand Side indicates the tooling operation that the LHS has captured.

It is also important to note that under certain cases the tool type selection also depends on the decisions made on the machine selection in the previous rule set. For example, if the water jet is chosen in rule set 4, then only the tool of type water jet is an applicable option in rule set 5. Further, if a drilling machine is chosen in rule set 4, then all kinds of milling tools are excluded from the feasible options in rule set 5. This casual constraint between the two rule sets further limits the size of the search tree.

After a tooling rule has been selected and applied to the current sub-volume, this sub-volume is marked as machined. Then the representation propagates to rule sets 6 and 7, which are “post-processing” rule sets. These sets generally take care of any remaining issues regarding label changes and graph modifications after each sub-volume has been manufactured. For example, a very often recognized rule in rule set 7 is to add new “original” labels to those faces which become exposed to the tool after the current sub-volume has been removed.

5.3 OVERALL DESCRIPTION OF THE DESIGN PROCESS

After rule sets 6 and 7, one complete step in a manufacturing plan has been defined to machine a given sub-volume. Then the representation will go back to rule set 1 to start another loop for another sub-volume. This representation loop is shown in Figure 13. If all the sub-volumes for a given part are machinable, the similar loop for each sub-volume will continue until all sub-volumes are machined. At that time, since there are no

more non-machined sub-volume for rule set 1 to choose, the looping process will terminate with a complete manufacturing plan for the part, which we refer to as a goal in the search tree.

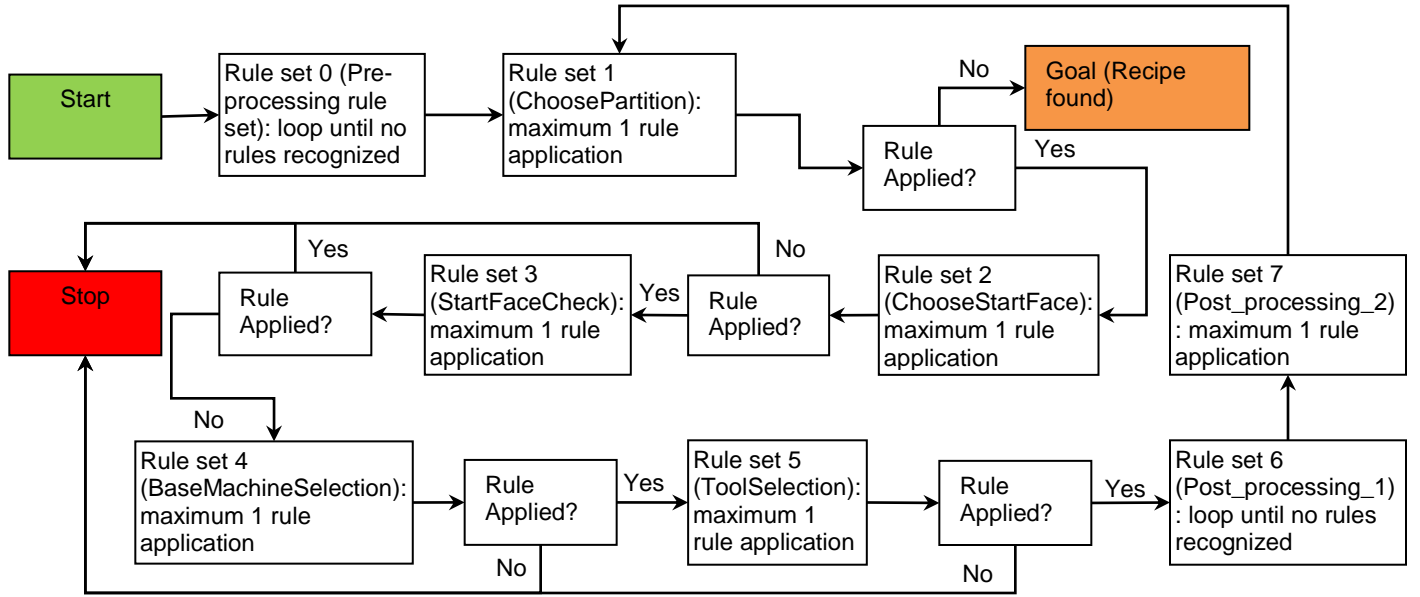


Figure 13: Flowchart of rule sets

However, as seen in Figure 13, rather than finding a complete machining recipe, the reasoning process can also end at different stopping points defined by different rule sets. Depending on the function each rule set performs, the representation loop ends at these stop points either when there is no rule applied in certain rule sets (rule sets 2, 4 and 5), or when a particular termination rule is recognized (rule set 3). For the first scenario, for example, the looping process will stop if there is no tooling rule in the Tool Selection rule set that is recognized for a sub-volume. If this case happens, the user can gain an insight that a current sub-volume of a given part is actually not manufacturable with current available tooling operations described in the Tool Selection rule set. Since a

foundry's capability is always mapped into different tooling rules, it is equivalent to conclude that the manufacturing of this part is beyond given foundry's capabilities. One can either redesign the part to make it easier to manufacture, or one can import more advanced machines and tools into the foundry to cover required tooling operations. For the second scenario, for example, the rules in rule set 3 define several infeasible cases of tool entry face selection. If any rule in rule set 3 is invoked, the tool entry face selected in rule set 2 becomes invalidated, and the search will terminate after this rule has been applied.

Therefore, depending on the given part and the knowledge of the manufacturability analysis built in the rules, a complete looping process described in Figure 13 will either find a feasible manufacturing plan, or converge on no plan. One should be aware that this complete reasoning process only represents one branch in the search tree in Figure 1. The whole search process actually contains numerous branches where each branch is depicted as one complete looping process in Figure 13. More detailed discussion about the search tree is given in next.

6. Heuristic Search for Process Planning

Now that the seed graph and all necessary rules have been described, a search process using these rules is presented which seeks feasible and optimal manufacturing plans. A graph grammar based software tool called GraphSynth [21] previously developed by the research team is used as a platform where the seed graph can be loaded and a Recognize-Choose-Apply cycle is invoked to define the tree. During the search a candidate host graph is provided to a recognition procedure which checks all the rules within a single rule set to find valid rules that can successfully change the graph. This defines a list of options which are essentially different branches in a search tree (similar to Figure 1). Amongst these options, one is chosen. However, given the expanse of computer memory most tree-search algorithms choose all possible paths thus defining a population of states. The Apply procedure executes the L-to-R graph transformation algebra to change the host state into a new graph.

In several of the rule sets described in Section 5 (0, 3, 6, and 7), the options do not define meaningful alternative paths. The rules within these rule sets simply prepare, fix, or check qualities of the graph. Many of the resulting options are confluent which means that they result in graph changes which do not negate other options from being called. As such these rule sets do not define alternative decisions in the tree. However, the decision rule sets (1, 2, 4, and 5) create decisions based on (1) the sub-volume to be removed, (2) the tool entry face where machining is initiated, (4) the type of machine and fixture orientation to use, and (5) the choice of tool to use. Therefore, the alternative options each of these rule sets recognizes constitute the branches of the search tree as shown similarly in Figure 1.

With GraphSynth, we implemented a depth-first search of the tree of valid manufacturing plans. During the search, the rules were used to define the successors at each state in the tree. When no successors are identified through the tree-search, then the resulting plan is either complete or a dead end. Dead end candidates are discarded and completed plans are stored and evaluated.

The search algorithm uses various pruning techniques. First, we are not concerned with inferior plans, because we wish to display only the best process plans to the designer. To this end we implemented functions to support detection of Pareto-dominated process plans. Such plans, when detected, are simply pruned from the search which reduces the number of plans the user has to sift through as well as improves the performance of our solver. Second, we detect duplicate sub-problems and cache intermediary results. The search algorithm has access to information about which sub-volumes must be machined, and in several cases, many of these sub-volumes are relatively independent with one another. A naive algorithm would explore all possible permutations of processing these sub-volumes. Instead, we implemented a powerful duplicate detection technique to reduce the size of these permutations. For a given subset of sub-volumes that have been machined, our algorithm remembers the particular subset, so that if we encounter the same subset in a future search sub-tree, we can simply skip over it. This technique resulted in over 30 and 40-fold speedups in several of our test cases, including test parts provided by Rolls-Royce, making representative challenge parts solvable within under a minute.

We also implemented two methods of obtaining a more diverse sampling of the various process plans in the search space. Especially in cases where we are unable to completely enumerate all process plans, we at least present to the user a set of plans that adequately cover the search space. As an improvement to our best-first search algorithm,

we introduce randomization when generating children nodes during the search to ensure a diverse sample. We have also implemented beam search, another search algorithm that complements our best-first search algorithm, which also features diverse sampling of the search space. Due to this sampling, the logic behind the Pareto domination algorithms was much more complicated. In particular, we had to compute the convex hull over a set of points, and used that for testing Pareto domination. Because of the many additional calculations, we ran into the problem of floating-point rounding errors, and therefore had to extend our code to explicitly handle these precision issues.

We have pulled in an open source planner and scheduler, called Fast Downward [22], and have code that translates a sub-problem in our domain into PDDL (Planning Domain Definition Language), the general planning language. This allows our search algorithm to delegate the sub-problem of choosing which machining plan to use for each sub-volume as well as the scheduling of such plans to the dedicated solver. Our encoding includes accessibility constraints, which captures which faces of which sub-volumes are now accessible due to machining operations on other sub-volumes.

Additionally, our search algorithm captures the hierarchical way that we reason about machining various sub-volumes, which resulted in significantly faster performance compared to a solver without the technique, allowing us to handle larger and complex parts in a shorter amount of time. This addresses the problem where a search algorithm might reason over all potential subsets of machining sub-volumes in the desired part in conjunction with reasoning over how to machine each individual sub-volume. Considering both problems simultaneously multiplies the difficulty of one with the other. By first enumerating all possible ways of machining each sub-volume individually, we can provide all such options to a dedicated and optimized scheduler which will pick and

reorder such options very quickly. This decoupling of the two problems significantly reduces the time to find a solution.

7. Examples and Discussion

In this section, we aim to disclose the new ideas, heuristics, and characteristics of the representation tool that are hidden in the specially designed seed lexicon and grammar rules through detailed case studies. These case studies will illustrate how the completeness of the search space generated from the representation is validated, the heuristics used to recover all the necessary manufacturing operations for a feature-interacted machining region, and the tool's adaptivity to both traditional and non-traditional manufacturing techniques. After that, the scope, capability and constraints of the representation tool will be clearly depicted. The comparison between the current representation tool with the other commercial software showing the differences and advantages of our tool in terms of manufacturing feedback analysis will be given at the end of this section.

7.1 DESCRIPTION OF TESTED CASES

The current grammar representation has been tested on over 20 solid models, ranging from self-designed simple geometries to complex suspension components provided by the research partners. In this section, three examples are provided, including one easy part (the shape shown as part A in Figure 1), one real part, which is a chassis part for small vehicle (Figure 16), and one complex part, which is also a component assembled in a small vehicle and is referred to as “radio box” (Figure 18). For the simple example, the generated search space that consists of all the feasible manufacturing plans will be verified. For the main chassis and the radio box, selected recipes for these parts and the underlying characteristic of the representation tool will be explained

7.1.1 Validation of the Completeness of the Representation Space

For the simple part, one may easily come up with a manufacturing plan. One such simple plan may be to first remove the bottom cuboid from the bottom face ha1 (as indicated in Figure 3), then drill the top hole from face ha7. In this way, only one fixture is needed. However, in a complete search tree created by automatically employing the rules on this seed graph, this plan is only one branch. All other branches representing different alternative plans are simultaneously generated. One sample plan is shown in Figure 14.

```
Candidate #2: time = 63.11:57:38.9580000
Drill, UMC, End_Mill, UMC, Solution_Obtained
f0: 0
f1: NaN
labels: Drill, UMC, End_Mill, UMC, Solution_Obtained
recipe:
StartFace( f_n9 ; ; ha9)
drill_2( f_n9 n8 n1 ; a17 ; ha9 Partition_1.STEP ha8)
VertexMachined( n8 ; ; Partition_1.STEP)
FindPartitionHyperarc( ; ; Partition_1.STEP)
ThreeAxisMachine( f_n12 f_n9 ; f_a_9_12 ;)
FaceToBeOriginal( f_n7 n8 ; ; ha7 Partition_0.STEP)
ConvexShapeMachinedOld( ; ; Partition_1.STEP)
StartFace( f_n1 ; ; ha1)
mill_1_forPaper( f_n1 f ; ha1)
ConvexShapeMachined( f_n1 n0 ; ; ha1 Partition_0.STEP)
VertexMachined( n0 ; ; Partition_0.STEP)
VertexMachined( n1 ; ; Partition_0.STEP)
VertexMachined( n2 ; ; Partition_0.STEP)
VertexMachined( n3 ; ; Partition_0.STEP)
VertexMachined( n4 ; ; Partition_0.STEP)
VertexMachined( n5 ; ; Partition_0.STEP)
VertexMachined( n6 ; ; Partition_0.STEP)
VertexMachined( n7 ; ; Partition_0.STEP)
VertexMachined( n8 ; ; Partition_0.STEP)
VertexMachined( n9 ; ; Partition_0.STEP)
VertexMachined( n10 ; ; Partition_0.STEP)
FindPartitionHyperarc( ; ; Partition_0.STEP)
ThreeAxisMachine( f_n13 f_n1 ; f_a_1_13 ;)
OriginalFaceRemoved( f_n0 n0 ; ; ha0 Partition_0.STEP)
OriginalFaceRemoved( f_n2 n7 ; ; ha2 Partition_0.STEP)
OriginalFaceRemoved( f_n7 n8 ; ; ha7 Partition_0.STEP)
ConvexShapeMachinedOld( ; ; Partition_0.STEP)
goalcheck( ; ;)
Candidate Manufacturing Plan 2:
-step 1: fixed face: f_n12, tool entry face: ha9, use machine type: UMC, with to
ol type: Drill, to remove volume of partition: Partition_1.STEP
-step 2: fixed face: f_n13, tool entry face: ha1, use machine type: UMC, with to
ol type: End_Mill, to remove volume of partition: Partition_0.STEP
```

Figure 14: A sample manufacturing plan for the solid model shown as part A in Figure 1

In this manufacturing plan, the upper portion is a step-by-step description of how the rules were implemented. An executable machining plan was provided at the end. Based upon this recipe, two steps are needed to make the part. First step is to fix the bottom face and drill the top hole from its top surface. In this case, a VMC is needed to finish this operation. Then the part is flipped over and the top face is fixed. The remaining cuboid is end milled from its bottom surface. Of the 96 valid plans that are

found (as discussed next), there are several inefficient plans like this one, and several of the more likely single-fixture approach.

```
number of solutions found: 96
Nodes: 869
CPU Time: 00:00:01.3416086
Nodes/Sec: 647.729896782117
Goal Tests: 127
```

Figure 15: Summary report of the search space for the part discussed in Figure 1

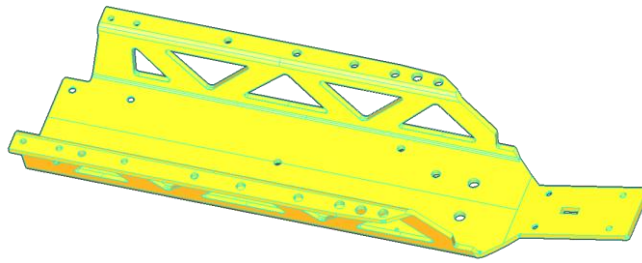
Figure 15 is a summary report after all possible manufacturing plans have been found. Within 2 seconds 96 solutions were generated for this simple part. The completeness is reflected by the total nodes in the search tree while the efficiency is reflected by the search rate, i.e. Nodes/Sec. One may doubt the large size of solutions. However, this number can actually be verified by a pure mathematical derivation.

If starting from ha1, only the End Milling tool is capable of removing the bottom cuboid. Despite the three bounding box faces that are coplanar with the outer surface of this cuboid where machining will happen, there are three other faces available for fixturing. After removing the cuboid, the hole can be either drilled or end milled from either its top face or its inner planar face. For the first case, there are five different fixtures, and for the latter one, there are six. So in total, $1 \times 3 \times (2 \times 5 + 2 \times 6) = 66$ solutions are found. However, the upper face, ha9 can also be chosen as the first entry face. Both End Milling tool and drill bit can be used to remove the hole. Five bounding box faces are available for fixture. After the hole, the cuboid can only be End Milled from ha1 with three different fixtures. So another $2 \times 5 \times 1 \times 3 = 30$ solutions are found. Therefore, in total 96 candidate plans are derived, exactly the same number of solutions found by the software.

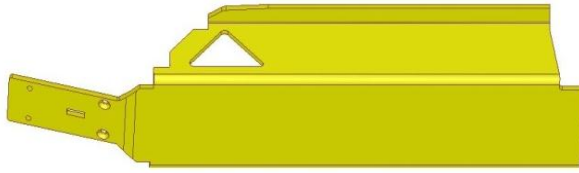
This example shows the validity of the grammar representation scheme. The grammar rules are good at creating a complete search space, which covers all possible and realistic candidate manufacturing plans for a given part, while the geometric reasoning about feature interactions is properly conducted by a specially designed search sequence for the rule sets.

7.1.2 Non-Traditional Manufacturing Processes in the Representation

Another example given is a main chassis part for a small autonomous vehicle (Figure 16). This part is unique due to its inclined tip, one rectangular through pocket and counter-sunk holes. Apparently, it is cut from sheet metal with subsequent bending operations. In addition, non-traditional machining operations are needed to cut out sharp corners for the pocket. For simplicity, some duplicate holes and triangular pockets were removed. However, even the simplified part (Figure 16(b)) has 12 sub-volumes, which represent at least 12 steps to machine this part. Due to the complexity, the seed graph converted from the simplified model is not shown.



(a) Original main chassis part



(b) Simplified main chassis part

Figure 16: The main chassis part

For this part, the preferred manufacturing plan is to first cut out the sheet metal with exactly the same shape, then drill out the holes and cut out the pockets, after that machine all countersinks continuously, and finally do several bending operations. However, since the software first looks at the final part then does a backward reasoning of how it was machined, one may expect that bending operations, instead of being the last few operations, are actually replaced by unbending operations at the very beginning of a generated manufacturing plan. For the same reason, the seed graph used in this case was not converted from the simplified part, but from the unbent sheet metal part. As mentioned in section 4, rather than removing material, bending operations significantly change the shape of a given part. As a result, the right bounding box and negative part could not be generated from that simplified part (Figure 16(b)). To solve this problem, a special heuristic is designed in the Convex Decomposition such that once a bending operation is detected for a given part, all needed pre-unbending operations are executed

in the convex decomposition first, then the unbent part is used to generate the seed graph. In this case, the seed graph was converted from a sheet metal part which has already been unbent from the simplified main chassis part. A sample recipe is shown in Figure 17.

```
Candidate Manufacturing Plan 1:
-step 0: Bending Operation detected, Pre_unbend this part for 5 times
-step 1: fixed face: f_n108, tool entry face: ha95, use machine type: UMC, with
tool type: WaterJet, to remove volume of partition: Partition_10.STEP
-step 2: fixed face: f_n107, tool entry face: ha94, use machine type: UMC, with
tool type: WaterJet, to remove volume of partition: Partition_9.STEP
-step 3: fixed face: f_n107, tool entry face: ha90, use machine type: UMC, with
tool type: WaterJet, to remove volume of partition: Partition_8.STEP
-step 4: fixed face: f_n108, tool entry face: ha84, use machine type: UMC, with
tool type: Drill, to remove volume of partition: Partition_7.STEP
-step 5: fixed face: f_n107, tool entry face: ha83, use machine type: UMC, with
tool type: CounterSink, to remove volume of partition: Partition_6.STEP
-step 6: fixed face: f_n108, tool entry face: ha77, use machine type: UMC, with
tool type: Drill, to remove volume of partition: Partition_5.STEP
-step 7: fixed face: f_n107, tool entry face: ha76, use machine type: UMC, with
tool type: CounterSink, to remove volume of partition: Partition_4.STEP
-step 8: fixed face: f_n107, tool entry face: ha72, use machine type: UMC, with
tool type: WaterJet, to remove volume of partition: Partition_3.STEP
-step 9: fixed face: f_n107, tool entry face: ha54, use machine type: UMC, with
tool type: WaterJet, to remove volume of partition: Partition_2.STEP
-step 10: fixed face: f_n107, tool entry face: ha3, use machine type: UMC, with
tool type: WaterJet, to remove volume of partition: Partition_0.STEP
-step 11: fixed face: f_n107, tool entry face: ha29, use machine type: UMC, with
tool type: WaterJet, to remove volume of partition: Partition_1.STEP
-step 12: fixed face: f_n107, tool entry face: ha104, use machine type: UMC, with
h tool type: WaterJet, to remove volume of partition: Partition_11.STEP
```

Figure 17: A sample manufacturing plan for the chassis part

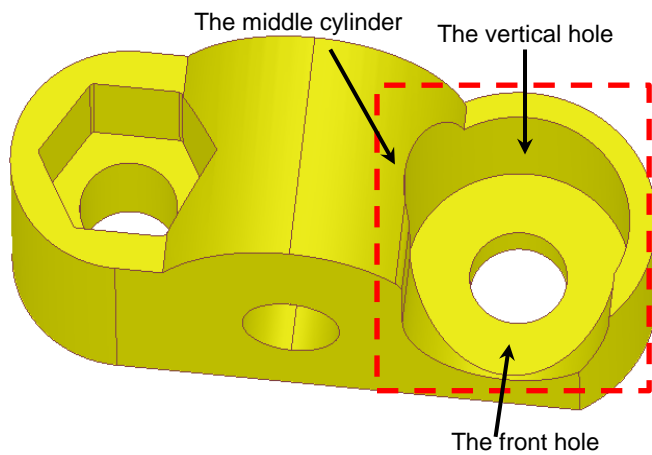
From the recipe one can see that actually five unbending operations are needed before machining process starts. After that, the Water Jet is used to cut most of the sub-volumes. It is very important to note that a counter sink operation always occurs after a drilling operation to satisfy the manufacturing constraint that there is no hole before the hole is drilled.

The complete search space of all possible manufacturing plans for this part expands exponentially as the total number of sub-volumes increases. For the simplified part with 12 sub-volumes (Figure 16(b)), an estimation of the size of the search space is conducted from the fact that there are at least one tool entry face, two fixtures and machine types and three tooling operations available, which in total represents 6 different options, for each sub-volume to choose. Moreover, these 12 sub-volumes can be manufactured in a random order. Therefore, in total at least $6^{12} \times 12! \gg 20$ trillion solutions are included in the search space. This underscores the need for an approach to eliminate as many of the poor solutions as possible to focus on more beneficial solutions.

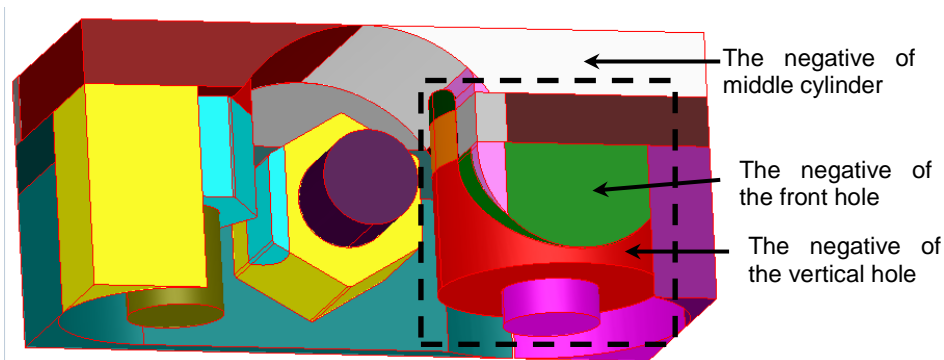
An ongoing evaluation research for the optimization of the search space generated in the representation is discussed in next and the details are given by Blarigan, et al [23].

7.1.3 Heuristics in the Representation for Feature Interactions

The third example, the radio box solid model, is given to illustrate the representation tool's capability of extracting sub-volumes from the interacted regions in the solid using specially designed heuristics. From Figure 18 (a) we can see that on the right hand side of this part, there is a complex region (shown in a red box), which is an interaction of a vertical hole, a front hole and a cylindrical shape in the middle (as indicated by arrows). The negative shape for this region is indicated in Figure 18 (b). Starting from this negative shape, our tool first detects all the features in this region by reasoning about the properties of the negative geometric elements, like the curved edges and the cylindrical faces. Then the special heuristic guides the convex decomposition to cut the region such that the most complete features are recovered. After that, the other small regions are decomposed based upon the other features they belong to. More details about this heuristic in the convex decomposition are given in [20].



(a) The original part of radio box



(b) The decomposed compound negative solid of the radio box

Figure 18: The solid model of radio box

Note that the other features can only be partially recovered from the remaining regions. This is due to nature of feature interactions: by removing one complete feature, the other features will be unavoidably damaged. However, the partially recovered features are important not only because the necessity of generating separate manufacturing steps to remove them, but also in that these features provide guidance/clues to the representation tool in deciding how to machine them. These clues

are extracted by the representation during the reasoning process about the geometric elements and their properties of the incomplete features.

```

Candidate Manufacturing Plan #12:
-step 1: fixed face: f_n156, tool entry face: ha67, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_9.STEP
-step 2: fixed face: f_n158, tool entry face: ha55, use machine type: HMC, with
tool type: End_Mill, to remove sub-volume: Partition_8.STEP
-step 3: fixed face: f_n155, tool entry face: ha51, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_7.STEP
-step 4: fixed face: f_n154, tool entry face: ha42, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_6.STEP
-step 5: fixed face: f_n156, tool entry face: ha33, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_5.STEP
-step 6: fixed face: f_n158, tool entry face: ha32, use machine type: HMC, with
tool type: End_Mill, to remove sub-volume: Partition_4.STEP
-step 7: fixed face: f_n156, tool entry face: ha152, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_22.STEP
-step 8: fixed face: f_n158, tool entry face: ha148, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_21.STEP
-step 9: fixed face: f_n158, tool entry face: ha145, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_20.STEP
-step 10: fixed face: f_n158, tool entry face: ha20, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_2.STEP
-step 11: fixed face: f_n156, tool entry face: ha19, use machine type: HMC, with
tool type: End_Mill, to remove sub-volume: Partition_3.STEP
-step 12: fixed face: f_n158, tool entry face: ha142, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_19.STEP
-step 13: fixed face: f_n157, tool entry face: ha21, use machine type: WaterJet,
with tool type: WaterJet, to remove sub-volume: Partition_17.STEP
-step 14: fixed face: f_n158, tool entry face: ha115, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_16.STEP
-step 15: fixed face: f_n156, tool entry face: ha110, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_15.STEP
-step 16: fixed face: f_n158, tool entry face: ha107, use machine type: HMC, with
tool type: End_Mill, to remove sub-volume: Partition_18.STEP
-step 17: fixed face: f_n158, tool entry face: ha58, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_14.STEP
-step 18: fixed face: f_n156, tool entry face: ha87, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_13.STEP
-step 19: fixed face: f_n153, tool entry face: ha84, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_12.STEP
-step 20: fixed face: f_n158, tool entry face: ha79, use machine type: HMC, with
tool type: End_Mill, to remove sub-volume: Partition_11.STEP
-step 21: fixed face: f_n158, tool entry face: ha71, use machine type: UMC, with
tool type: End_Mill, to remove sub-volume: Partition_10.STEP
-step 22: fixed face: f_n156, tool entry face: ha8, use machine type: HMC, with
tool type: End_Mill, to remove sub-volume: Partition_1.STEP
-step 23: fixed face: f_n157, tool entry face: ha6, use machine type: HMC, with
tool type: Ball_Mill, to remove sub-volume: Partition_0.STEP
Time: 230.7236328272, Cost: 145.940276694561

```

Figure 19: A sample manufacturing plan for the radio box

For this radio box case, the final decomposed volumes are shown in Figure 18 (b). The red sub-volume is the negative shape of the vertical hole; the green sub-volume is the negative of the front hole; and the white sub-volume is the negative of the middle cylinder. Obviously the convex decomposition decides on the vertical hole as the most complete feature to recover. This tells the representation that a drilling or a milling operation is needed to remove this sub-volume. Moreover, the front hole is partially recovered as the green sub-volume. Although this is not a complete feature, its semi-circular front face and the partial cylindrical face provide enough information to inform the representation that another drilling or milling operation is needed to machine this green sub-volume. As for the white sub-volume, it consists of all the remaining negative

material for the interacted region. Since the white sub-volume is prismatic, the representation tool always calls the milling tool to remove it. Overall, based on the number of features detected, a feasible sequence of manufacturing steps that are needed to remove the entire interacted region is generated. Similarly, the representation implements the reasoning process for the interacted region on the left as well as the other sub-volumes of the radio box. A complete manufacturing plan is provided in Figure 19. In this plan, step 17 removes the white sub-volume; the green sub-volume is removed in step 20, and step 22 removes the red sub-volume.

This heuristic implemented in our tool to deal with the interacted machining regions has a beneficial side effect. By recovering the most complete features first, it simulates a roughing manufacturing process in the foundry, which is used to remove as much material as possible in the first few operations. The following operations for the remaining small or partial features generated in the representation tool represent a finishing process. More precise tooling operations are needed in order to satisfy the high tolerancing requirements that are associated with the positive part. As a result, less amount of material is expected to be removed.

It is also important to be aware that in this manufacturing plan most of the sub-volumes are removed by the same type of tool. Even for the red sub-volume, instead of using a drilling tool, an end milling tool is called to remove this sub-volume. The advantage of minimizing the types of tools needed in one manufacturing plan is that the manufacturing time and cost can be reduced. Without changing the tools, many refixtures become unnecessary and the manufacturing process becomes more continuous and faster. Additionally, given the fact that the milling tool is more efficient than the drilling tool in terms of removing a large amount of material, the milling tool is always preferred in the current representation as long as the tolerance requirements for each sub-volume are

satisfied. However, the drawback is that no geometric dimensioning and tolerancing information is considered during the representation process. Such information becomes pivotal in deciding the manufacturing recipe in the highly precise manufacturing industry. Currently we still limit our tool in circumstances like one-off workshops, or three axis machining centers. But more future research is being and going to be conducted towards the parametric reasoning during the manufacturing process. More detailed discussion is given in section 8.

7.2 CHARACTERISTICS OF IMPLEMENTATION AND DEPLOYMENT

The current representation serves as the fundamental component of AMFA, the Automated Manufacturing Feedback Analysis system. Figure 20 shows the web-based GUI of AMFA. In this system, the representation serves as the foundation to automatically reason about a CAD model and to define detailed and optimal manufacturing plans. In this section, the testing report based upon AMFA is provided and the characteristics of implementation and deployment are summarized.

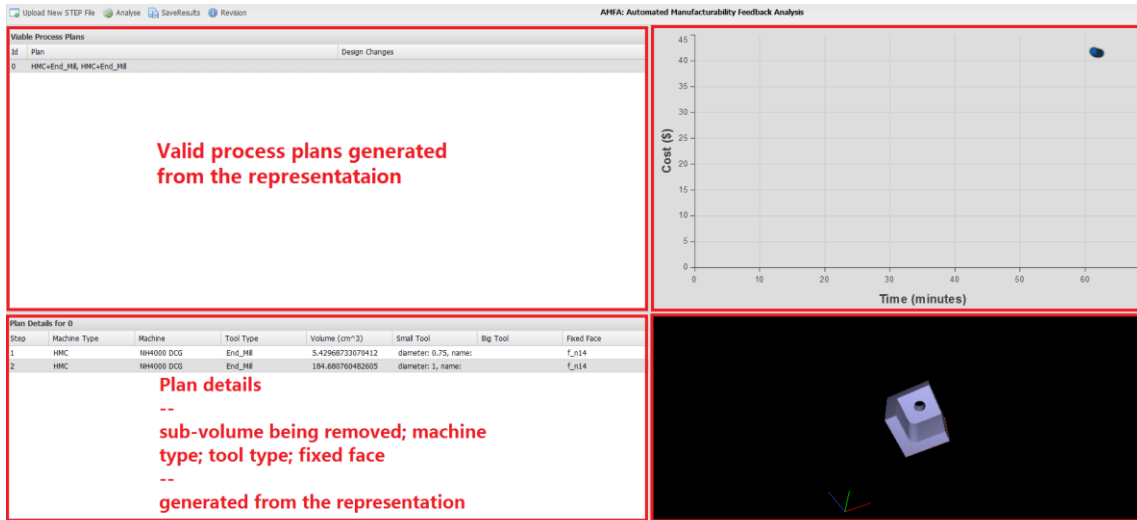


Figure 20: AMFA GUI Web Page

Some of the features of the representation are as follows:

- The representation tool is customizable based on available resources of a particular foundry. The manufacturing capability of the representation ranges from the widely used traditional tooling operations to the non-traditional and new-born manufacturing techniques. We have demonstrated that the tool can work with manufacturing machine libraries provided by various entities.
- The representation tool is proficient in taking care of both simple and complex geometries, including feature interactions. The tool is able to decompose the complex regions into simple and prismatic machining features for the rules to reason about.
- Through the GUI, the representation rules are able to show recommended design changes to the designer to improve manufacturability when a non-manufacturable part is uploaded.
- The representation tool allows the user to step through the individual steps of a proposed plan and see the staging models: the geometry that is created after any

prefix of the plan has been executed. The trick is that each one step in the manufacturing plan removes one sub-volume in the compound negative solid. The remaining negative solid constitutes the concurrent staging model. This feature allows easy validation of a plan, and provides a concrete means for designers to communicate with manufacturing engineers about the feasibility and pros and cons of feasible plans.

7.3 MANUFACTURING PROCESSES HANDLED

Currently the representation tool is mainly focusing on the 3D axis machining process. Most of the traditional manufacturing operations that can be implemented in 3D axis machines are encoded into the tool. They include:

- **Milling**

Both end milling and ball milling are handled. While end milling is intended for use in most cases, such as profile milling (pockets, holes, slots, chamfers, etc.), face milling, tracer milling or plunging, ball milling is used specifically for creating inner fillets for a given part.

- **Drilling**

The drilling operation in the representation is able to produce both through holes and blind holes. It is a highly generalized manufacturing process that indicates all possible regions (typically holes) in a given part where a drill bit can be used.

- **Counter-sinking**

This is a specific finishing process for the countersinks sitting on the holes. It represents a manufacturing process to produce countersinks with specially designed tools.

- **Counter-boring**

In the representation, this manufacturing process is encoded into milling operations, i.e., for the counter-bores, instead of using specific tools, the representation uses all available milling tools to machine them.

- Bending

This is a non-material-removal manufacturing process. The representation implements this operation by first detecting all regions that need to be bent, then calling bending operation to take care of these regions.

- Welding

For a given final part, the representation detects if the welding operations have ever been used to manufacture it. If such operation is detected, an unbending operation will be called to separate the welded parts, and each separate part will then be manufactured using the operations mentioned above.

- Manufacturing processes from a casted part

If a part needs to be machined from a casted raw material, the representation will use the casted model as the bounding box and subtract the input solid model from the bounding box to get the regions to be machined. These regions will be manufactured using the operations mentioned above.

A few non-traditional manufacturing processes are also built and being built into the representation tool.

- Sheet metal cutting (Water-jet cutting)

For sheet metal parts, instead of machining it traditionally, the representation tool uses water-jet cutting to cut out the part profile.

7.4 PART GEOMETRIES THAT CAN BE PROCESSED

Despite that the representation tool has been tested on a variety of parts provided from Rolls-Royce, Pennsylvania State University (PSU) and other research partners, there are a few constraints that the parts to be uploaded must satisfy. The constraints are summarized into several categories and a few screen shoots of the sample parts for each category are provided.

1) The parts should be able to be decomposed into prismatic partitions using the convex decomposition algorithm. The sample part Figure 37, although looks complex, is implementable by AMFA since all its partitions are prismatic.

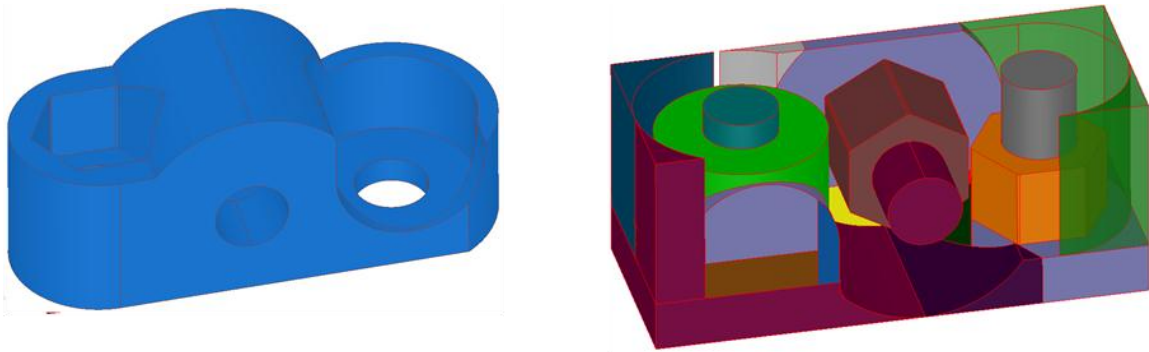


Figure 21: Sample part 1—the Radiobox

2) There should be no fillets/round edges around the boundaries of the parts. Due to the limitation of the CAD kernel used in the convex decomposition, small fillets/round edges will crash the software. Fortunately, since these features are machined in the final finishing process, removing them does not affect the generation of high level manufacturing recipes.



Figure 22: Sample parts 2 and 3

One research effort has been implemented to do automatic de-featuring for the input solid model if some unwanted features were detected. The aim is to inform the users what aspects need to be modified for the uploaded part in order not to crash the tool, and to make the representation tool more reliable and user-friendly. Currently the capability of detecting round edges around the boundary of the original solid and automatically removing them has been built into the software. However, this functionality only works for simple edges. When several edges interact or the edge is of a complex shape, the knowledge built in the tool will not recognize them. However, suggestions and diagnostic report for this part will be provided to the users instantly such that the designer can easily fix the problems.

3) All kinds of sheet metal parts are implementable in AMFA. Amongst these parts, some may need non-traditional operations, like bending and welding. The representation tool will generate corresponding recipes for these operations. The main chassis part discussed previously is an example for this.

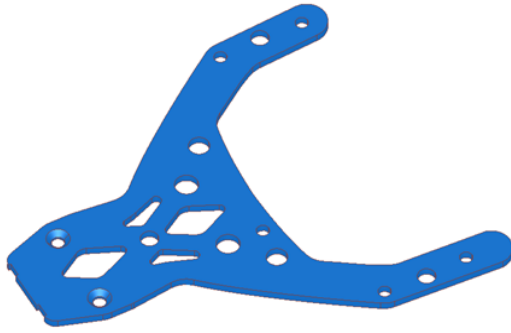


Figure 23: Sample parts 4

4) AMFA is able to process pre-casted parts. As seen in Figure 24 and Figure 25, part (a) is the final model that is to be created, and part (b) is the pre-casted model that is used as the raw material to create part (a). The representation tool is able to distinguish the part machined from a casted model from the part that is traditionally machined from the raw material, and generate different manufacturing plans correspondingly. For the main chassis part discussed previously and shown in Figure 16 and Figure 25, if the user uploads both the final part (Figure 25 (a)) and the raw material, the pre-casted model (Figure 25 (b)), to the software, the convex decomposition will automatically compute the negative removal volumes from the pre-casted model and pass the new seed graph to the representation such that a user-specified manufacturing plan (Figure 26) can be generated correctly.

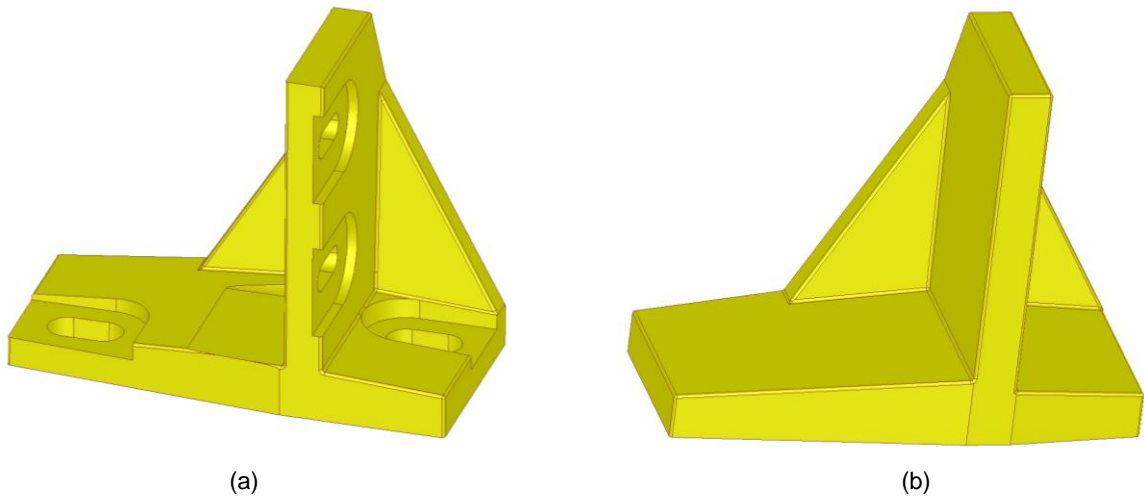


Figure 24: Sample part 5—A pre-formed Bradley Vehicle Bracket

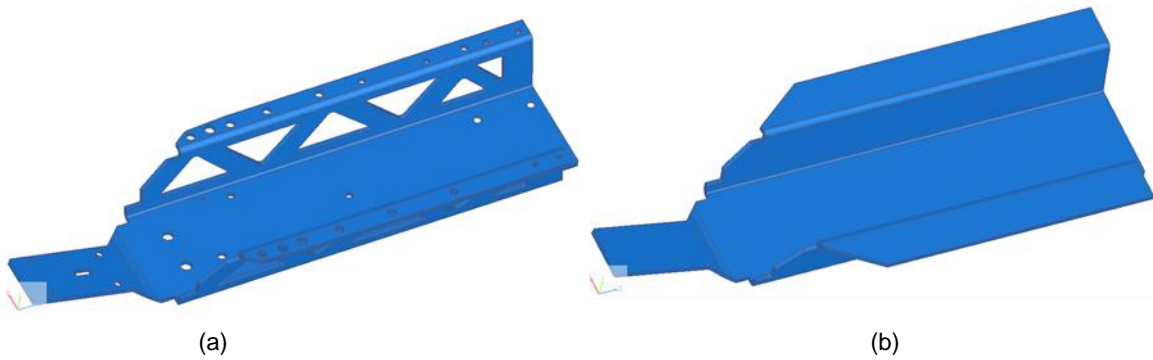


Figure 25: Sample part 6—A pre-formed chassis part

By comparing the new plan with the one in Figure 17 we see the differences are that: 1) there are no more bending operations detected; 2) to remove the same sub-volume, the fixed face and the machine type used may be different due to different shapes of raw material used in the two cases.

```

Candidate Manufacturing Plan #52:
-step 1: fixed face: f_n97, tool entry face: ha51, use machine type: UMC, with tool type: End_Mill, to remove sub-volume: Partition_9.STEP
-step 2: fixed face: f_n97, tool entry face: ha47, use machine type: WaterJet, with tool type: WaterJet, to remove sub-volume: Partition_8.STEP
-step 3: fixed face: f_n97, tool entry face: ha43, use machine type: UMC, with tool type: End_Mill, to remove sub-volume: Partition_7.STEP
-step 4: fixed face: f_n97, tool entry face: ha39, use machine type: WaterJet, with tool type: WaterJet, to remove sub-volume: Partition_6.STEP
-step 5: fixed face: f_n97, tool entry face: ha35, use machine type: WaterJet, with tool type: WaterJet, to remove sub-volume: Partition_5.STEP
-step 6: fixed face: f_n97, tool entry face: ha31, use machine type: WaterJet, with tool type: WaterJet, to remove sub-volume: Partition_4.STEP
-step 7: fixed face: f_n88, tool entry face: ha27, use machine type: WaterJet, with tool type: WaterJet, to remove sub-volume: Partition_3.STEP
-step 8: fixed face: f_n84, tool entry face: ha23, use machine type: WaterJet, with tool type: WaterJet, to remove sub-volume: Partition_2.STEP
-step 9: fixed face: f_n97, tool entry face: ha57, use machine type: WaterJet, with tool type: WaterJet, to remove sub-volume: Partition_10.STEP
-step 10: fixed face: f_n89, tool entry face: ha9, use machine type: WaterJet, with tool type: WaterJet, to remove sub-volume: Partition_1.STEP
-step 11: fixed face: f_n101, tool entry face: ha4, use machine type: HMC, with tool type: Ball_Mill, to remove sub-volume: Partition_0.STEP
Time: 79.0420472039447, Cost: 34.6893191647513

```

Figure 26: A new manufacturing plan for the pre-formed chassis part

7.5 VALIDATION

A set of tests for all the sample parts discussed previously have been implemented in FeatureCAM [24] as well. The manufacturing plans generated were analyzed and compared with that provided by our representation tool. The results show our technique's expertise in instantly generating manufacturability feedbacks for a given part and effectively communicating them with the end users.

First, we take part A in Figure 1 as an example to illustrate the manufacturing plan generated from featureCAM, which is shown in Figure 27.

Operation List						
<input checked="" type="radio"/> Automatic Ordering <input type="radio"/> Manual Ordering						
R	Operation	Feature	Tool	Feed	Speed	Depth
▶	finish	face1	facemillM3200	117.00 IPM	5200 RPM	
	rough pass 1	side1	endmillM2100:reg	49.66 IPM	3003 RPM	2.250 in.
	finish	side1	endmillM2100:4...	91.67 IPM	4620 RPM	2.250 in.
	spotdrill	hole1	SD_90_M2000	14.32 IPM	1256 RPM	0.380 in.
!	drill	hole1	*****	0.00 IPM	0 RPM	0.750 in.
	rough pass 1	side2	endmillM2500:reg	49.66 IPM	2523 RPM	0.750 in.
	finish	side2	endmillM2500:reg	45.84 IPM	3881 RPM	0.750 in.
Results						

Figure 27: A sample manufacturing plan generated from featureCAM for part A in Figure 1

In this manufacturing plan, seven steps are required to machine this part. The first step is to smooth a bounding box face from which the front cuboid will be machined. The second and the third steps represent the rough and finish passes needed to remove the negative cuboid. Then the hole is drilled in two steps, including one rough pass using spot drill and one finish pass. The last two steps employ an end mill tool to remove any chips left around the boundary of the final part. Along with each step, detailed parametric information regarding the tool diameter, feed rate, spindle speed and the feed depth is provided. The processing-specific sub-steps for each machining region makes this manufacturing plan very technical and practically implementable.

Comparatively, the recipe given from the representation tool as listed in Figure 14 is a very high-level description of how the manufacturing process for each machining region can be initiated. Only two steps are included in this plan. However, the information contained in each step differs from what we have seen in the plan from featureCAM quite a lot. Rather than being specialized and hard to understand, this simplified plan conveys the fundamental but important details for each manufacturing operation, including where to start, how to fix the part, what machine type to use, and which region to remove, to the general users, if not the manufacturing engineers. By

reading this plan, the end user can quickly and clearly get an insight about how each machining region can be manufactured. If any aspect, which could be no tool entry face, no face to fix, no tool to use, or no machine type available, causes the failure of processing a region, the user can easily come up with a specific design modification that heals the non-manufacturable aspect. However, one should note that manufacturability analysis in the representation is implemented by the rules through the topological, rather than parametric, reasoning processes. At this stage, we intentionally relax the parametric constraints imposed by the foundry limitations. One reason is that these constraints, such as the lack of appropriate size of tool, are usually not valid. There is no necessity to sacrifice the efficiency of the topological manufacturability analysis in order to incorporate the parametrical reasoning for very few applicable cases. The other reason is that the parametrical constraints may violate the designer' intention. Since the design specifications should always be prioritized, the violated constraints are considered secondarily in our tool.

For example, let us go back to the manufacturing plan in Figure 27. In front of the fifth step of this plan, there is a red exclamation mark, which indicates an error in this plan because that the hole in the given part is too small for featureCAM to find a suitable drilling tool to remove it. While this error is valuable to signal the foundry limitation, it might unnecessarily break down the simulation from the designer's perspective since the dimension of the hole might be intentionally specified in order to meet certain engineering constraints and the designer do not want to do anything about it. In this case, featureCAM might give a wrong message to the designer that this part is not manufacturable, which would undoubtedly slow down the design efficiency that the software aims to improve.

In order to fill in the gap between what the typical manufacturability analysis tool provides and the information the end users really need, our representation tool switches its focus onto the topological reasoning. The underlined logic is that if a part is topologically not manufacturable because of: 1) inaccessible regions in this part (for example, the inner sharp corners in part a in Figure 11); 2) geometric defects in the given solid model; 3) unrecognized/invalid geometric elements; etc, then this part would always be non-manufacturable unless required redesign is made by the users.

For instance, consider the part A in Figure 11. It is apparently not manufacturable due to an inaccessible sharp edge from either too feed direction. From the manufacturing plan in Figure 28, we see that the representation tool asserts the same conclusion in almost no time. Additionally, through the AMFA GUI (Figure 20), the user is informed that all the inner sharp edges should be removed before the part can be machined. This example shows how our tool communicates with the user instantaneously in terms of design improvements.

```

0 1.1: ChoosePartition( ; ; Partition_0.STEP>
1 2.1: StartFace( f_n0 ; ; Partition_0.STEP ha0>
1 2.1: StartFace( f_n1 ; ; Partition_0.STEP ha1>
1 2.1: StartFace( f_n2 ; ; Partition_0.STEP ha2>
2 4.1: BaseFaceVMC( f_n8 f_n2 ; f_a_2_8 ;>
2 4.2: BaseFaceHMC( f_n6 f_n2 ; f_a_2_6 ;>
2 4.2: BaseFaceHMC( f_n10 f_n2 ; f_a_2_10 ;>
2 4.1: BaseFaceVMC( f_n10 f_n1 ; f_a_1_10 ;>
2 4.2: BaseFaceHMC( f_n6 f_n1 ; f_a_1_6 ;>
2 4.2: BaseFaceHMC( f_n8 f_n1 ; f_a_1_8 ;>
2 4.1: BaseFaceVMC( f_n6 f_n0 ; f_a_0_6 ;>
2 4.2: BaseFaceHMC( f_n8 f_n0 ; f_a_0_8 ;>
2 4.2: BaseFaceHMC( f_n10 f_n0 ; f_a_0_10 ;>
number of solutions found: 0
Nodes: 14
CPU Time: 00:00:00.5772037
Nodes/Sec: 24.2548687750962
Goal Tests: 0
Times Pruned: 0
Max Cache Size: 0
Max Queue Size: 5

```

Figure 28: A sample manufacturing plan generated from the representation for part A in Figure 11

The same part has also been checked in featureCAM. To compare, featureCAM proposed a manufacturing plan for this part, which is shown in Figure 29.

Operation List						
<input checked="" type="radio"/> Automatic Ordering <input type="radio"/> Manual Ordering						
R	Operation	Feature	Tool	Feed	Speed	Depth
→	finish	face1	facemillM3200	2971.8 MPPM	5200 RPM	
	rough pass 1	side1	endmillM2500.reg	1261.3 MPPM	2523 RPM	35.000 mm
	finish	side1	endmillM2500.reg	1164.3 MPPM	3881 RPM	35.000 mm
⚠	rough pass 1	side2	endmillM2500.reg	1261.3 MPPM	2523 RPM	35.000 mm
⚠	finish	side2	endmillM2500.reg	1164.3 MPPM	3881 RPM	35.000 mm
Results						

Figure 29: A sample manufacturing plan generated featureCAM for part A in Figure 11

In this plan, we see that due to the sharp edges, two rounds of rough and finish passes were employed to remove the negative cuboid. However, no matter how high the manufacturing precision that the tool can achieve, the sharp edges would never be created exactly as what they were designed. Figure 30 indicates the chips left in the final shape after the manufacturing plan has been implemented. We see that there are still a lot of remaining areas that the tool cannot access. These areas are actually where the non-

manufacturable sharp edges are located. Rather than trying to tackle the bad geometries by the ineffective employment of the foundry capability, our representation tool returns the redesign suggestions directly to the user in an early phase. This feature distinguishes the representation tool as a user-centric instant and effective manufacturability feedback provider.

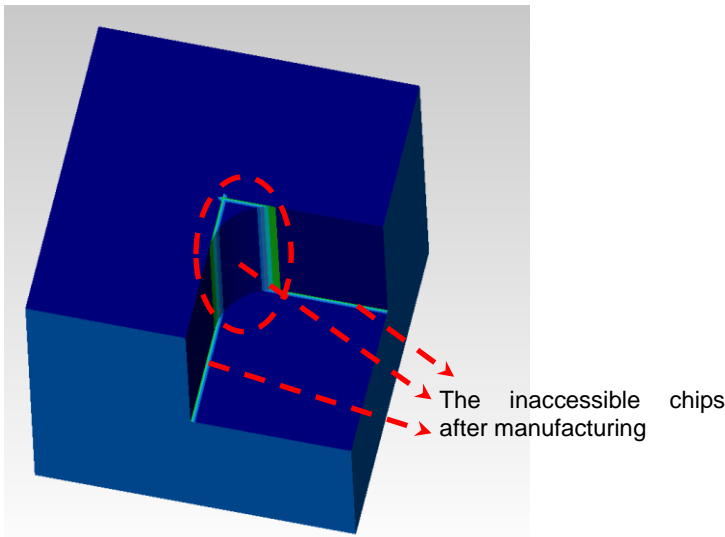


Figure 30: The comparison between the original part A in Figure 11 and the solid machined in featureCAM

Moreover, the topological analysis implemented in the representation tool automatically provides additional information that featureCAM does not cover. Such information includes the face to fix (setup design), the type of machine to choose, and the amount of material to remove for each option. In featureCAM, the user needs to manually specify the setups and most probably the user needs to redesign the setup if the current fixture is not applicable. Instead, our tool generates all feasible setups for each operation in a single search process and different users can choose different fixtures based upon their own criteria. The type of machine to use for each operation is important in that many operations might be able to be implemented in a single machine. For example, a

milling machine can also do drilling operation. Therefore, the final manufacturing plan consisting of no matter how many steps can always be implemented in much fewer machine types. By being aware of this information, the user is able to reduce the final manufacturing time and cost a lot. The amount of material to be removed for each operation is a message that helps the user track each manufacturing step. By visually displaying the staging model of the part after each operation the representation tool provides a user-friendly interface from which the user can easily interact with the tool in a much more efficiency manner. For example, by inspecting the staging models, the user is able to inform the tool that certain features must be removed in the last a few operations due to the tolerance requirements and the tool can adjust the output manufacturing plan accordingly.

Another pivotal feature that distinguishes the representation tool from the others is that our tool is fully automatic. The only input from the user is a valid solid model. After that, the tool will automatically finish the manufacturability analysis in a moderate speed. The output is either a space of all possible manufacturing plans, or the feedback regarding the recommended changes that need to be made in order to manufacture the part. This feature further guarantees the capability of the representation tool of providing instant manufacturability feedback.

8. Conclusion and Future Work

In this thesis we described a graph grammar based approach to reasoning about the manufacturability of given solid models. The goal of this tool is to automatically assess whether a part can be manufactured as the designer specifies, or provide the designer feedback about what part of the design specification cannot be met given the constraints of a foundry. In addition, we described in detail the constituent pieces of innovations that collectively make up this representation tool. A two-module structure of this approach separates the geometric conversion of a solid model, which is done in the convex decomposition, and the topological reasoning of manufacturability, which is realized in the representation. While the convex decomposition is mainly used to decompose a model into multiple sub-volumes, which serve as the basis for the representation, it can also take advantage of the numerous functionalities built in Open CASCADE to pre-process some complex parts, as in the second example which un-bends a part as required. In this way, more realistic manufacturing plans can be generated.

On the representation side, a topological reasoning process for a solid model is realized using the seed lexicon and graph grammar based rules. By storing all geometric elements and topological relations of a solid model in the seed lexicon through nodes, arcs, hyperarcs and labels, the generic nature of the representation scheme is ensured. The completeness and validity of the representation are realized by the rules and a specially designed search sequence for the rule sets.

However, it is not enough to only have a large number of automatically generated manufacturing plans. A continuing effort in evaluating these plans and deciding the optimal one based on user-specified objective functions is occurring. To implement this effort, a more detailed evaluation and optimization process is being developed. The

output of the representation, including the tool type, machine type, fixture for each operation in a manufacturing plan, is used to evaluate each plan. The search process then makes a decision amongst these evaluated plans. The goal is to find the optimal manufacturing plans. This work has been presented in detail by Blarigan, et al [23].

Another research effort currently in progress is to extend tooling operations and machine types to turning centers (i.e. lathe) and multi-axis milling centers. For complex parts, a user may prefer a manufacturing plan with few steps which can be implemented in a multi-axis milling center rather than a manufacturing plan that is decomposed into lots of sub-steps and can be implemented with a 3-axis milling machine. More non-cutting machining and tooling operations need to be translated into rules to handle such complexities.

The missing piece for a complete automation of the design and manufacturing process is the coupling between the design phase and the manufacturing phase. Today these coupling issues are typically resolved through multiple design-build-test-redesign iterations leading to longer schedules, capital-intensive manufacturing costs, reduced reliability and limited reconfiguration and reuse capability of the manufacturing enterprise. AMFA provides this missing concurrent engineering capability by bringing real world manufacturing constraints forward into concurrent design cycle, by identifying manufacturing constraints of a foundry, and thereby generating the candidate manufacturing plans for the users to choose.

With an ongoing cooperation with the evaluation and optimization and a continuous work on expanding the capability of the representation scheme, an intelligent graph grammar based tool, which not only provides design insights in terms of manufacturability to the designers, but also outputs optimal manufacturing plans to the engineers, is close to becoming a reality.

References

- [1] E. L. Russell, Automated manufacturing planning, New York: American Management Association, 1967.
- [2] H. B. Marri, A. Gunasekaran and R. J. Grieve, "Computer-Aided Process Planning: A State of Art," *The International Journal of Advanced Manufacturing Technology*, vol. 14, 1998.
- [3] H. R. Berenji and B. Khoshnevis, "Use of artificial intelligence in automated process planning," vol. 5, no. 2, 1986.
- [4] G. Chryssolouris and S. Chan, "An Integrated Approach to Process Planning and Scheduling," *Annals of the CIRP*, vol. 34, no. 1, 1985.
- [5] H.-p. Wang and R. A. Wysk, "A knowledge-based approach for automated process planning," *International Journal of Production Research*, vol. 26, 1988.
- [6] R. Sharma and J. Gao, "Implementation of step 224 in an automated manufacturing planning system," 2002.
- [7] R. D. Allen, J. A. Harding and S. T. Newman, "The application of STEP-NC using agent-based process planning," vol. 43, pp. 655-670, 2005.
- [8] H. Ehrig, G. Engels, H.-J. Kreowski and G. Rozenberg, Handbook of Graph Grammars and Computing by Graph Transformations, NJ: World Science Publication Co., 1999.
- [9] F.-L. Krause, "Knowledge integrated product modeling for design and manufacturing," 1989.
- [10] M. P. a. W. C. R. JungHyun Han, "Manufacturing Feature Recognition from Solid Models: A Status Report," *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, vol. 16, no. 6, 2000.
- [11] J. Shah, "An assessment of features technology," *Computer-Aided Design*, vol. 23, no. 5, 1991.
- [12] J. J. Shah, D. Anderson, Y. S. Kim and S. Joshi, "A Discourse on Geometric Feature Recognition From CAD Models," vol. 123, March 2001.
- [13] M. W. Somashekar Subrahmanyam, "An overview of automatic feature recognition techniques for computer-aided process planning," *Computers in Industry*, vol. 26, pp. 1-21, 1995.
- [14] M. M. a. R. L. Kashyap, "Geometric reasoning for recognition of 3-D object features," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 949-965, 1990.
- [15] M. M. Timo Laakko, "Feature modelling by incremental feature recognition," *Computer-Aided Design*, vol. 25, no. 8, 1993.
- [16] S. J. a. T. C. Chang, "Graph based heuristics for recognition of machined features from a 3-D solid model," *Computer-Aided Design*, vol. 20, pp. 58-66, 1988.
- [17] S. N. T. a. R. L. Kashyap, "Geometric reasoning for extraction of manufacturing features in iso-oriented polyhedrons," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 16, no. 11, p. 1087-1100, 1994.
- [18] Y. S. Kim, E. Wang, C. S. Lee and H. M. Rho, "Feature-Based Machining Precedence Reasoning and Sequence Planning," Atlanta, 1998.
- [19] C. Ertelt and K. Shea, "An application of shape grammars to planning for CNC machining," San Diego, California, 2009.

- [20] A. Eftekharian and M. I. Campbell, "Convex Decomposition of 3D Solid Models for Automated Manufacturing Process Planning Applications," in *ASME International Design Engineering Technical Conferences IDETC*, Chicago, IL, USA, 2012.
- [21] M. I. Campbell, "GraphSynth2: software for generative grammars and creative search," 7 2 2012. [Online]. Available: <http://graphsynth.com/>.
- [22] M. Helmert, "The Fast Downward Planning System," *Journal of Artificial Intelligence Research*, vol. 26, no. 1, 2006.
- [23] B. V. Blarigan, M. I. Campbell, A. Eftekharian and T. Kurtoglu, "Automated Estimation of Time and Cost For Determining Optimal Machining Plans," in *ASME International Design Engineering Technical Conferences IDETC*, Chicago, Illinois, USA, 2012.
- [24] Delcam, "FeatureCAM," [Online]. Available: <http://www.featurecam.com/index.asp>. [Accessed 25 04 2012].

Vita

Wentao Fu was born in Henan Province, China. After graduating from Shanghai Jiao Tong University (SJTU) with a Bachelor's degree in Mechanical Engineering and Automation in 2010, he entered the Mechanical Engineering Graduate Program at The University of Texas at Austin in August, 2010. He is currently a research assistant in the Automated Design Lab held by Dr. Matthew I. Campbell. His research focuses on the design optimization, and more specifically a graph grammar based approach to automated manufacturing planning.

Permanent Email: fwtay001@gmail.com

This thesis was typed by the author.